UNIVERSITY OF NOVI SAD
FACULTY OF TECHICAL SCIENCES

Bojan Marinković

Interconnection of Heterogeneous Overlay Networks: Definition, Formalization and Applications

DOCTORAL THESIS

Novi Sad
2014

УНИВЕРЗИТЕТ У НОВОМ САДУ ● ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Redni broj, RBR:	
Identifikacioni broj, IBR:	
Tip dokumentacije, TD:	Monografska dokumentacija
Tip zapisa, TZ:	Tekstualni štampani materijal
Vrsta rada, VR:	Doktorska disertacija
Autor, AU:	Bojan Marinković
Mentor, MN:	Dr Zoran Ognjanović, naučni savetnik
Naslov rada, NR:	Povezivanje heterogenih prekrivajućih mreža: definicija, formalizacija i primene
Jezik publikacije, JP:	Engleski
Jezik izvoda, JI:	Srpski/Engleski
Zemlja publikovanja, ZP:	Srbija
Uže geografsko područje, UGP:	Vojvodina
Godina, GO:	2014
Izdavač, IZ:	Autorski reprint
Mesto i adresa, MA:	Univerzitet u Novom Sadu, Fakultet tehničkih nauka
Fizički opis rada (poglavlja/str./ citata/tab./sl./graf./priloga), FO:	6/153/85/9/5/16/1
Naučna oblast, NO:	Matematika; informatika
Naučna disciplina, ND:	Teorijsko računarstvo; Matematička logika
Ključne reči, PO:	Prekrivajuće mreže Distribuirane heš tabele Verifikacija Mašine apstraktnih stanja Temporalne logike Digitalizacija kulturne baštine
UDK:	378.1
Čuva se, ČU:	Biblioteka Fakulteta tehničkih nauka
Važna napomena, VN:	
Izvod, IZ:	Doktorska disertacija se bavi temama vezanim za prekrivajuće mreže, njihovom definicijom, formalizacijom i primenama. Dati su opisi *Chord* i *Synapse* protokola korišćenjem ASM formalizma, kao i dokaz korektnosti formalizacije Chord protokola na visokom nivou, kao i njegovo profinjenje. Izvršena je verovatnosna ocena uspešnosti pretrage pomoću Synapse protokola. Prestavljena je ažurirana verzija Predloga sheme meta podataka za pokretna kulturna

	dobra, kao i Predlog sheme meta podataka za opis kolekcija. Implementiran je Distribuirani katalog digitalizovanih kolekcija kulturne baštine Srbije zasnovan na Chord protokolu.
Datum prihvatanja teme od NN veća, DP:	
Datum odbrane, DO:	
Članovi komisije, KO: Predsedik:	Dr Silvia Ghilezan, redovni profesor, Fakultet tehničkih nauka Univerziteta u Novom Sadu
Član:	Dr Luigi Liquori, Research Director, INRIA - Francuski nacionalni institut iz oblasti računarstva i automatike
Član:	Dr Miroslav Popović, redovni profesor, Fakultet tehničkih nauka Univerziteta u Novom Sadu
Član:	Dr Veljko Milutinović, redovni profesor, Elektrotehnički fakultet Univerzita u Beogradu
Mentor:	Dr Zoran Ognjanović, naučni savetnik, Matematički institut SANU
Potpis mentora:	

UNIVERSITY OF NOVI SAD ● FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, ANO:	
Identification number, INO:	
Document type, DT:	Monograph type
Type of record, TR:	Printed text
Contents Code, CC:	PhD thesis
Author, AU:	Bojan Marinković
Mentor, MN:	Zoran Ognjanović, Ph.D., Research Professor
Title, TI:	Interconnection of Heterogeneous Overlay Networks: Definition, Formalization and Applications
Language of text, LT:	English
Language of abstract, LA:	Serbian/English
Country of publication, CP:	Serbia
Locality of publication, LP:	Vojvodina
Publication year, PY:	2014
Publisher, PU:	Author's reprint
Publ. place, PP:	University of Novi Sad, Faculty of Technical Sciences
Physical description (ch./pg./ ref./tab./pict./graphs/app.), PD:	6/153/85/9/5/16/1
Scientific field, SF:	Computer Science; Mathematics
Scientific discipline, SD:	Theoretical Computer Science; Mathematical Logic
Key words, UC:	Overlay Networks Distributed Hash Tables Verification Abstract State Machines Temporal Logic Digitization of the Cultural Heritage
UC:	378.1
Holding data, HD:	The Library of the Faculty of Technical Sciences, Novi Sad
Note, N:	
Abstract, AB:	This Ph.D. thesis addresses topics related to overlay networks, their definition, formalization and applications. Descriptions of the Chord and Synapse protocols using the ASM formalism is presented, and both a high-level and a refined proof of the correctness of the Chord formalization is given. A probabilistic assessment of the

	exhaustiveness of the Synapse protocol is performed. An updated version of the Proposal of metadata schemata for movable cultural heritage as well as a Proposal of metadata schemata for describing collections are provided. Based of the Chord protocol, a Distributed catalog of digitized collections of Serbian cultural heritage is implemented.
Accepted by the Scientific Board, ASVB:	
Defended, DE:	
Thesis defend board, DB: President:	Silvia Ghilezan, Ph.D., Full Professor, Faculty of Technical Sciences, University of Novi Sad
Member:	Luigi Liquori, Ph.D., Research Director, INRIA, France
Member:	Miroslav Popović, Ph.D., Full Professor, Faculty of Technical Sciences, University of Novi Sad
Member:	Veljko Milutinović, Ph.D., Full Professor, School of Electrical Engineering, University of Belgrade
Mentor:	Zoran Ognjanović, Ph.D., Research Professor, Mathematical Institute of the Serbian Academy of Sciences and Arts
Menthor's sign:	

Acknowledgements

The story of this thesis begun in late 1998, during my high school days, when I got a new course on programing languages and a new teacher. After a couple of months he decided to pick several students from my class and to give them non-classical problems to try to solve them. Zoran Ognjanović was the professor and I had an opportunity to become the member of this group. This was the very beginning of my collaboration with Zoran and now, the time has come to thank him for the inspiration and new ideas for all this time.

The next important steps for this thesis happened during 2008 and 2009 with the TEMPUS project DEUKS. The new PhD program started as a collaboration of MISANU and FTN. This was the first time that I met "girls from Novi Sad", as we like to call them, led by professor Silvia Ghilezan. I would like to thank them all for all the beautiful memories from DEUKS summer school, RDP 2011 conference and other events. Also, this project gave me an opportunity to spend three splendid months at one of the most beautiful places on Earth - The Côte d'Azur, to join the professor Luigi Liquori's team in INRIA and to enter the world of distributed programming. Thank you Luigi, for all the support I had then, and all the collaboration we had in the meanwhile.

I would also like to thank professors Miroslav Popović and Veljko Milutinović for doing me the honour of being members of the jury for my thesis.

Last, but definitely not the least, my biggest thanks goes to my friends and, especially, my family for being the foundation pillar of my life.

Hvala, hvala, hvala.

Abbreviations Used in the Thesis

ASM	-	Abstract State Machine
DBMS	-	Database Management System
DDBMS	-	Distributed Database Management System
DHT	-	Distributed Hash Table
IS	-	Information System
NRDBMS	-	Non-Relational Database Management System
P2P	-	Peer-To-peer
TTL	-	Time-To-Leave

Abstract

This thesis presents the definition of the Synapse protocol — a meta protocol for connecting heterogeneous overlay networks; its formal specification together with the specification of the already existing overlay network protocol Chord; a proof of the correctness of the Chord protocol using the ASM formalism; a refined version of this proof that makes use of a novel inference procedure for dealing with a time flow isomorphic to ω^2 and finely granulated ASM rules; a probabilistic assessment of the properties of the Synapse protocol; an updated version of the proposal of metadata schemata for describing digitized objects and collections for cultural and scientific heritage of Serbia; and an implementation of a catalog of digitized collections based on the Chord protocol.

Overlay networks — a class of P2P systems — are structures that provide an organizational overview of the resources from a logical standpoint. They are fully independent of the underlying network and the connections between devices on the physical level. Overlay networks have been identified as a promising model that could cope with the current issues of the Internet, such as scalability, resource discovery, failure recovery, routing efficiency, and, in particular, in the context of information retrieval.

The structure of this thesis is as follows: first, an overview of the Chord protocol, the ASM formalism, and the state-of-the-art in the digitization practices are presented. This is followed by the description of the Chord protocol using the ASM formalism, and the proof of the correctness of the Chord protocol using high-level rules. Since it is problematic to assume that the transition time in ASM is equal to 0, as one could then be able to infer that the system is in two different states at one time instant, a new temporal logic with time flow isomorphic to ω^2 is introduced. After that, using the finely granulated ASM rules of the Chord protocol and the new inference system, revisited proofs of the previously proved theorems are provided.

The next part of the thesis contains the definition of the Synapse protocol, its formal specification using the ASM formalism — an extension of the already presented Chord specification, and a probabilistic assessment of the properties of the Synapse protocol, which are then compared to the results obtained from extensive simulations and experiments.

The final part of the thesis contains an updated version of the proposal of metadata schemata for describing digitized objects and collections for cultural and scientific heritage of Serbia, accompanied by the description of the main principles behind and an implementation of a catalog of digitized collections based on the Chord protocol.

The results presented in this thesis can serve as a foundation for future research. The demonstrated techniques can be re-used to obtain results for more network protocols. The correctness proofs can constitute a starting point for formal verification using a formal proof assistant, so as to get a formally verified proofs of the correctness of the Chord protocol. The semantics of the introduced temporal logic can be fully adopted into the ASM inference system. On the other hand, the probabilistic methods used can be applied to solving further problems in graph theory. Finally, the implemented catalog can be a good testbed for a new concept of distributed NRDBMSs.

Rezime

Ova teza daje definiciju protokola Sinapsa, meta protokola za povezivanje heterogenih prekrivajućih mreža, formalnu specifikaciju postojećeg protokola za prekrivajuće mreže *Chord*-a, kao i protokola Sinapsa, dokaz korektnosti *Chord* protokola korišćenjem formalizma mašina apstraktnih stanja, kao i profinjenih dokaza koji koriste proceduru odlučivanja u kome je tok vremena izomorfan skupu ω^2, kao i granulisana ASM pravila, verovatnosnu ocenu očekivanih osobina protokola Sinapsa, ažuriranu verziju predloga shema metapodataka za opis digitalizovanih objekata i kolekcija kulturne i naučne baštine u Srbiji, kao i implementaciju kataloga digitalizovanih kolekcija zasnovanog na *Chord* protokolu.

Prekrivajuće mreže, klasa P2P sistema, su struktura koja je nezavisna od mreža nad kojima su realizovane i koje zaiste povezuju same uređaje, i predstavlja logičku organizaciju resursa. One su prepoznate kao model mreža koji obećava koji može uspešno da se nosi sa trenutnim problema Interneta, kakvi su skalabilnost, otrkrivanje resursa, opravka od greške u radu, efikasnijeg rutiranja, i, posebno, u pogledu pretrage informacija.

Na početku ove teze dat je pregled *Chord* protokola, mašina apstraktnih stanja i trenutne situacije u procesima digitalizacije. Za ovim sledi opis *Chord* protokola korišćenjem formalizma mašina apstraktnih stanja. Dat je dokaz korektnosti *Chord* protokola korišćenjem pravila na višem nivou. S obzirom na potencijalni problem koji može da nastane u situaciji ako se pretpostavi da je vreme potrebno za izvršavanje tranzicija u formalizmu mašina apstraktnih stanja jednak 0, zbog činjenice da se u jednom trenutku sistem može naći u istovremeno u dva različita stanja, uvodi se nova temporalna logika sa vremenskim tokom izomorfnim skupu ω^2. Nakon toga, korišćenjem granulisanih pravila višeg nivoa i novog sistema izvođenja, daju se novi dokazi prethodno dokazanih teorema o korektnosti *Chord* protokola.

Sledeći deo sadrži definiciju protokola Sinapsa, formalnu specifikaciju koja koristi formalizam mašina apstraktnih stanja kao proširenje specifikacije *Chord* protokola i verovatnosnu ocenu očekivanih osobina ovog protokola, koje su porede sa rezultatima iscrpnih eksperimenata i simulacija.

Poslednji deo sadrži ažuriranu verziju predloga shema metapodataka za opis digitalizovanih objekata i kolekcija kulturne i naučne baštine u Srbiji. Za ovim sledi opis glavnih principa i realizacije kataloga digitalizovanih kolekcija zasnovanog na *Chord* protokolu.

Rezultati prezentovani u ovoj tezi mogu poslužiti kao osnova za buduća istraživanja. Tehnike korišćene ovde mogu biti upotrebljene na drugim mrežnim protokolima. Dobijeni dokazi mogu poslužiti za formalnu verfikaciju korišćenjem nekog interaktivnog dokazivača kako bi se dobili mašinski provereni dokazi korektnosti *Chord* protokola. Semantiku uvedene temporalne logike moguće je potpuno utopiti u sistem izvođenja mašina apstraktnih stanja. Nasuprot tome, korišćena metoda verovatnosne ocene može se primeniti u domenu teorije grafova. Na kraju, realizovani katalog može biti testno okruženje novog koncepta distribuiranih ne-relacionih sistema upravljanja bazama podataka.

Contents

Papers Included in the Thesis

The thesis is based on the following articles:

1. Z. Ognjanović, T. Butigan Vučaj and **B. Marinković**. *NCD Recommendation for the National Standard for Describing Digitized Heritage in Serbia.* In *Metadata and Semantics*, Edited by: Sicilia, Miguel-Angel; Lytras,Miltiadis D, Springer, pages 45–54, 2009.

2. L. Liquori, C. Tedeschi, L. Vanni, F. Bongiovanni, V. Ciancaglini and **B. Marinković**. *Synapse: A Scalable Protocol for Interconnecting Heterogeneous Overlay Networks.* In *Networking 2010*, Lecture Notes in Computer Science, vol. 6091 (p. 410), pages 67–82, 2010.

3. **B. Marinković**, L. Liquori, V. Ciancaglini and Z. Ognjanović. *A Distributed Catalog for Digitized Cultural Heritage.* In *ICT Innovations 2010*, CCIS 83 (p. 378), Edited by: Gusev, Marjan; Mitrevski, Pece, Springer-Verlag Berlin Heidelberg, pages 176–186, 2011.

4. **B. Marinković**, Z. Ognjanović and T. Butigan Vučaj. *A Distributed Implementation of a Catalog of Digitized Cultural Collections.* In *Review of the National Center for Digitization*, Issue 21, pages 19–24, 2012.

5. **B. Marinković**, Z. Ognjanović, D. Doder and A. Perović. *A Propositional Linear Time Logic with Time Flow Isomorphic to* ω^2. In *Journal of Applied Logic*, 12 (2), p. 208–229, 2014.

6. **B. Marinković**, V. Ciancaglini, Z. Ognjanović, P. Glavan, L. Liquori and P. Maksimović. *Analyzing the Exhaustiveness of the Synapse Protocol.* In *Peer-to-Peer Networking and Applications*, DOI: 10.1007/s12083-014-0293-z, 2014.

7. **B. Marinković**, P. Glavan and Z. Ognjanović. *Description of the Chord Protocol using ASMs Formalism. Under Review.*

1

Introduction

A decentralized P2P system [72] involves many peers (nodes) that execute the same software, participate with equal rights to the system, and can join or leave the system at any time. In such a framework, processes are dynamically distributed to the peers and there is no centralized control. P2P systems have no inherent bottlenecks and can potentially scale very well. Moreover, since there are no dedicated nodes critical for the functioning of the systems, those systems are more resilient to failures, attacks, etc., than their centralized counterparts. The main applications of P2P-systems involve file sharing, redundant storage, real-time media streaming, etc.

P2P systems are often implemented in the form of overlay networks [81], a structure independent of the physical organization of the underlying network. An overlay network provides a logical outlook on the organization of the resources. Recently, overlay networks have been identified as a promising model that could cope with the current issues of the Internet, such as scalability, resource discovery, failure recovery, routing efficiency, and, in particular, in the context of information retrieval.

Today, one can notice that not only many disparate overlay networks co-exist across the Internet, but that there also exist scenarios in which they compete for the same resources on shared nodes and underlying network links. One of the main problems of overlay networking is how to allow different overlay networks to interact and co-operate with each other. When it comes to the overlay networks that have already been developed, one can perceive a great extent of heterogeneity between them. In most cases, this heterogeneity renders them unable to co-operate, communicate, and exchange resources with one another without resorting to the costly, non-scalable, and security-compromising operation that is overlay merging.

On the other hand, there are many situations where different overlay networks could benefit from co-operation for various purposes, such as collective performance enhancement, larger shared information, better resistance to loss of connectivity (network partitions), improved routing performance in terms of delay, throughput, and packets loss (by, for instance, co-operative forwarding of flows). In the context of large-scale information retrieval, several overlays may wish to offer an aggregation of their resources to their potential common users, without relinquishing control over them. In terms of fault-tolerance, co-operation can increase the availability of the system. If one overlay were to

become unavailable, the global network would only undergo partial failure, as other different resources would still be usable.

Some of the overlay networks are realized in the form of DHTs that provide a lookup service similar to a hash table. A DHT consists of $\langle key, value \rangle$ pairs, and any participating peer can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the peers, in such a way that any change in the set of participants causes a minimal amount of disruption to the system. This allows a DHT to scale up to an extremely large number of peers and still be able to successfully handle continuous node arrivals, departures, and failures.

The Chord protocol [78–80] is one of the first, simplest, and most popular DHTs. The paper [78] which introduces Chord has been recently awarded the SIGCOMM 2011 Test-of-Time Award.

Recently, several NRDBMSs have been developed [21,54] that are commonly based on a Chord-like technology. To better analyze their behavior, it is useful to characterize scenarios in which the correctness of the underlying protocol holds. Following that idea, several deterministic conditions that guarantee correctness of Chord have been formulated in this thesis, and the corresponding statements have been proven. This contrasts the approaches from [56, 78–80] where a probabilistic analysis is proposed, and correctness holds with “high probability".

This characterization was realized by describing Chord using ASM [45] and proving the correctness of the formalization, motivated by the fact that the errors in concurrent systems are difficult to reproduce and are solely discovered through program testing. There are at least two reasons that justify the use of ASMs. First, ASMs are versatile machines that can simulate arbitrary algorithms in a direct and essentially coding-free way. Here, the term algorithm is taken in a broader sense, encompassing programming languages, architectures, distributed and real-time protocols, etc. The simulator is not supposed to implement the algorithm on a lower abstraction level; the simulation should be performed on the natural abstraction level of the algorithm. Second, a vast literature on ASMs shows how to model real-world complex systems closely and faithfully, and how to use models in order to verify that their properties hold (see, e.g. [12], the Bakery algorithm [10], the Railroad crossing problem [47], the Kerberos algorithm [8], Java formalization [14], [15], a special issue devoted to the method [11], etc). The ASM-code for Chord presented here has been written following one of the best implementations [29] of the high level C++-like pseudo code from [80]. The main difference between the implementation in [29] and this specification is in the addition of mechanisms proposed in [82] for improving the efficiency of detecting a failed successor which have been incorporated into the code.

The main objectives of Chord are maintaining the ring topology as nodes concurrently join and leave a network, mapping keys onto nodes, and distributed data handling. The formalism of ASM allows for a precise description of a class of possible runs — so called regular runs — of the protocol, and for the proof of correctness of the main operations with respect to these runs. Moreover, several examples of runs, given in Example 3.1.3, that violate the constraints for the regular runs illustrate how correctness can be broken in various scenarios.

There were only a few attempts to formally verify behavior of DHTs in general and Chord in particular [5, 6, 52, 56, 71, 82, 85]. They will be considered

below and will be compared with the approach presented in this thesis.

If it is assumed that transition time in ASMs is equal to 0, it is possible to infer that system is in two different states at one time, one state before and one state after the execution of the transition. This might lead to inconsistencies, especially in cases where time is crucial for the inference. For this reason, a temporal logic with time flow isomorphic to ω^2 is introduced, as well as the proofs of the same correctness theorems of the main Chord operations with the new semantics.

The ready-to-market DHT-based technology of structured overlay networks can be enriched with the capability of accessing different overlays through co-located nodes, i.e. peers who are, by the choice of the user, members of several overlays. Such nodes are not only able to query multiple overlays in order to find a match, but can also replicate requests, passing them through from one network to another, and efficiently collect results from these request. As a basic example, it is possible to consider two distant databases. A node of the first database stores a $\langle key, value \rangle$ pair. Without network cooperation, a node of the second database would not be able to communicate with the node of the first database and if it tries to search for that *key* it will never get the stored *value*. In another example, a number of nodes of an overlay network can get isolated by an underlay network failure, creating partitions within a single network. If some or all of those nodes could be reached via an alternative overlay network, then the partition could be recovered by using an alternative routing path.

In the context of large scale information retrieval, several overlays may want to offer an aggregation of their information/data to their potential common users without losing control of it. Imagine two companies wishing to share or aggregate information contained in their distributed databases, obviously while keeping their proprietary routing and their exclusive right to update it. Finally, in terms of fault-tolerance, cooperation can increase the availability of the system — if one overlay becomes unavailable the global network will only undergo partial failure as other distinct resources will be usable.

Having a single global overlay has many obvious advantages and is the *de facto* most natural solution, but it appears unrealistic in the actual setting. In some optimistic case, different overlays are suitable for collaboration by opening their proprietary protocols in order to build an open standard; in many other pessimistic cases, this opening is simply implausible for many different reasons (backward compatibility, security, commercial, practical, etc.). As such, studying protocols to interconnect collaborative (or competitive) overlay networks constitutes a potentially promising research vein.

A solution for interconnecting different overlay networks could be found in the use of a meta-protocol that allows a request to be routed through multiple heterogeneous networks, thus increasing the success rate of every request. One of the possible solutions for the inter-connection of heterogeneous overlay networks is the Synapse protocol [59], that is described in detail in Chapter 4. It is a generic and flexible meta-protocol that provides simple mechanisms and algorithms for easy interconnection of overlay networks. As an extension of the ASM formal description of the Chord protocol, a formal specification of Synapse within the formalism of ASM will be given. In addition to the specification of Synapse in ASM, a probabilistic estimate on the exhaustiveness of the Synapse protocol across a number of scenarios will be provided.

Digitization is an important step aimed in preservation and promotion of

heritage. It safeguards cultural diversity in the global environment and offers a rich treasure to the world-wide public of the Web. Usually, digitization can be seen as a collection of activities, including digital capture, transformation from analogue to digital form, description and representation of heritage objects and documentation about them, processing, presentation and long-term preservation of digitized content, etc.

In the digitization of catalog services, an IS has been shown to be essential in matching offers, requests, and resources. The IS is, in most cases, a front-end web site connected to a back-end database. A classical client-server architecture is usually sufficient to manage those services. Users register their profile onto one IS, and then post their offers/requests. In presence of multiple services, for technical and/or commercial reasons, it is not possible to share content across different ISs, despite the evident advantage. In most cases, ISs are not equipped with mechanisms for intercommunication concerning any of their features (lookup, search, etc.). Although, in principle, this does not affect the correct behavior of an IS, it is clear that interoperability would increase the overall quality of the service. Moreover, the classical shortcomings of client-server architectures make both services unavailable in case when one of them is down. Any attempt to make distinct and disconnected institutional client-server based architectures does not foresee any form of service interconnection, with the unpleasant consequence of losing potential matches between offers and requests between users of different communities on the same subject.

Also, the digitized documents are, by their nature, highly distributed resources. This triggered the development of the Catalog of digital collections of Serbia based on the Synapse protocol, as a real-life proof-of-concept.

The contributions of this thesis are:

- A description of the Chord and Synapse protocols using the ASM formalism
- Both high-level and refined proofs of correctness of the Chord formalization
- A temporal logic with time flow isomorphic to ω^2
- A probabilistic assessment of the exhaustiveness of the Synapse protocol
- An updated version of the Proposal of metadata schemata for movable cultural heritage given in [63]
- A proposal of metadata schemata for describing collections
- A realization of the application: Distributed catalog of digitized collections of Serbia

The remainder of the thesis is organized in the following way:

- Section 2.1 describes the Chord protocol in an “informal” way;
- Section 2.2 presents the basics of the ASM formalism;
- Section 2.3 gives an overview of the current digitization practices in Serbia and in the world;

- Chapter 3 introduces the ASM description of the Chord protocol with a high-level proof of the correctness of this specification, a temporal logic with time flow isomorphic to ω^2 and a refined proof of the correctness of the Chord protocol with respect to this time flow and low-level rules of its ASM description
- Chapter 4 contains the description of the Synapse protocol, its ASM specification and a probability assessment of its exhaustiveness;
- Chapter 5 presents two proposals of metadata standards for description of cultural and scientific heritage, and describes one realization of the Distributed Catalog of Digitized Collections of Serbia.

The thesis is concluded with a summary of the obtained results and an exposition of plans for further work, given in Chapter 6.

2

Background

2.1 Chord Protocol

As it is said, the Chord protocol [78–80] is one of the first, simplest and most popular DHTs. A number of nodes running the Chord protocol form a ring-shaped network. The main operation supported by Chord is mapping the given key onto a node using consistent hashing. The consistent hashing [51] provides load-balancing, i.e., every node receives roughly the same number of keys, and only a few keys are required to be moved when nodes join and leave the network. Each node in a Chord network (with N-nodes) needs "routing" information about only a few other nodes ($O(\log N)$), and resolves all lookups via $O(\log N)$ messages to other nodes. When the network is not stable, i.e., the corresponding "routing" information is out of date since nodes join and leave arbitrarily, the performance degrades. However, a recent development [82] has shown that Chord's stabilization algorithm (with minor modifications) maintains good lookup performance despite continuous failure and joining of nodes.

Identifiers are assigned to nodes and keys by the consistent hash function. The identifier for a node or a key, $hash(node)$ or $hash(key)$, is produced by hashing IP of the node, or the value of the key. The length of identifiers (for example m bits) must guarantee that the probability that two objects of the same type are assigned same identifiers is negligible. Identifiers are ordered in an identifier circle modulo 2^m. Then, the key key is assigned to the node such that $hash(node) = hash(key)$. If such a node does not exist, the key is assigned to the first node in the circle whose identifier is greater than $hash(key)$.

Every node possesses information on its current successor and predecessor nodes in the identifier circle. To accelerate the lookup procedure, a node also maintains routing information in the form of the so-called *Finger Table* with up to m entries. The i^{th} entry in the table at the node n contains the identifier of the first node s that succeeds n by at least 2^{i-1} in the identifier circle, i.e., $s = successor(n + 2^{i-1})$, where $1 \leqslant i \leqslant m$ (and all arithmetic is preformed modulo 2^m). The stabilization procedure implemented by Chord must guarantee that each node's successor pointer and finger table are up to date. The procedure runs periodically in the background at each node. To increase robustness, each Chord node can create a successor list of size r, containing the node's first r successors.

Beside the mapping of keys onto the set of nodes, the only other operations realized by Chord are adding/removing of a node to/from a network. When a node n joins an existing network, certain keys previously assigned to n's successor now become assigned to n. When node n leaves the network regularly, it notifies its predecessor and successor and reassigns all of its keys to the successor.

A node can search the network which runs the Chord protocol by executing following operations:

```
n.find_successor(id)
  if (id ∈ (n, successor])
    return successor;
  else
    n' = closest_preceding_node(id);
    return n'.find_successor(id);

n.closest_preceding_node(id)
  for i = m downto 1
    if (finger[i] ∈ (n, id))
      return finger[i];
  return n;
```

An example of such search is illustrated by Figure 2.1:

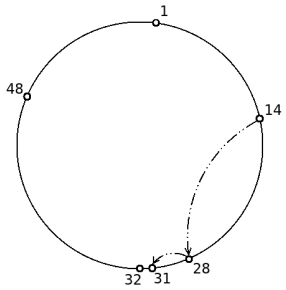


Figure 2.1: Function $find\_successor$

A node starts a new Chord network by:

```
n.create()
  predecessor = nil;
  successor = n;
```

A node can join the network which runs the Chord protocol by executing following operation:

```
n.join(n')
  predecessor = nil;
  successor = n'.find_successor(n);
```

Every node which is a member of a Chord network runs following operations periodically:

```
n.stabilize()
  x = successor.predecessor;
  if (x ∈ (n, successor))
```

```
      successor = x;
   successor.notify(n);

   n.notify(n')
      if (predecessor is nil or n' ∈ (predecessor, n))
        predecessor = n';

n.check_predecessor()
   if (predecessor has failed)
     predecessor = nil;

n.fix_finger()
   next = next + 1;
   if (next > m)
     next = 1;
   finger[next] = find_successor(n + 2^(next-1))
```

2.2 Abstract State Machines

The Abstract State Machines are defined in [12, 45, 46]. Here only the essential definitions are quoted.

A Gurevich's Abstract State Machine $\mathcal{A}$ is defined by a program *Prog* - consisting of a finite number of *transition rules*, at most countable set of states and initial states. $\mathcal{A}$ models the operational behavior of a real dynamic system $\mathcal{S}$ in terms of evolution of states.

A state S is a first-order structure over a fixed signature (which is also the signature of $\mathcal{A}$), representing the instantaneous configuration of $\mathcal{S}$. The value of a term t at S is denoted by $[t]_S$. The basic transition rule is the following function update

$$f(t_1, \ldots, t_n) := t$$

where f is an arbitrary n-ary function and $t_1, \ldots, t_n, t$ are first-order terms. To fire this rule in a state S evaluate all terms $t_1, \ldots, t_n, t$ at S and update the function f to $[t]_S$ on parameters $[t_1]_S, \ldots, [t_n]_S$. This produces another state S' which differs from S only in the new interpretation of the function f (since states represent memory, function update represents change in the content of one memory location).

Additionally, the following transition rules exist.

The *conditional constructor* produces "guarded" transition rules of the form:

```
if g then
  R1
else
  R2
endif
```

where g is a ground term (the guard of the rule) and R_1, R_2 are transition rules. To fire that new rule in a state S evaluate the guard; if it is true, then execute R_1, otherwise execute R_2. The `else` part may be omitted.

The *seq constructor* produces transition rules of the form:

```
seq
  R_1
  ...
  R_n
endseq
```

to apply $R_1, \ldots, R_n$ sequentially. Note that the *seq*-constructor is originally defined without the `endseq`-line, but it was added to improve readability.

The *par constructor* produces transition rules of the form:

```
par
  R_1
  ...
  R_n
endpar
```

to apply $R_1, \ldots, R_n$ simultaneously, when possible, otherwise do nothing.

If U is a universe name, v is a variable, $g(v)$ is a Boolean term and R is a rule then the following expression (*choose constructor*) is a rule with the main existential variable v that ranges over U and body R:

```
choose v in U satisfying g(v)
  R
endchoose
```

If there is an element $a \in U$ such that condition $g(a)$ is true, fire rule R (with a substituted for v), otherwise do nothing.

To express the simultaneous execution of a rule R for each x satisfying a given condition φ (*forall constructor*):

```
forall x with φ do
  R
endforall
```

A run (or computation) of $\mathcal{A}$ is a finite or infinite sequence $S_0; S_1; S_2; \ldots$ where S_0 is an initial state and every S_{i+1} is obtained from S_i executing a transition rule.

In general runs may be affected by the environment. Environment manifests itself via so-called external functions. Every external function can be understand as a (dynamic) oracle. The ASM provides the arguments and the oracle gives the result.

In a distributed Gurevich's Abstract State Machine $\mathcal{A}$ multiple autonomous agents cooperatively model a concurrent computation of $\mathcal{S}$. Each agent a executes its own single-agent program $Prog(a)$ as specified by the module associated with a by the function Mod. More precisely, an agent a has a partial view $View(a; S)$ of a given global state S as defined by its sub-vocabulary $Fun(a)$ (i.e. the function names occurring in $Prog(a)$) and it can make a move at S by firing $Prog(a)$ at $View(a; S)$ and changing S accordingly. The underlying semantic model ensures that the order in which the agents of $\mathcal{A}$ perform their actions is always such that no conflicts between the update sets computed for distinct agents can arise. The global program $Prog$ is the union of all single-agent programs. Nullary function Me, that allows an agent to identify itself among other agents, is interpreted as a for each agent a, and does not belong to

$Fun(a)$ for any agent a. It cannot be the subject of an update instruction and is used to parameterize the agent's specific functions. A sequential run of a distributed Gurevich's Abstract State machine $\mathcal{A}$ is a (finite or infinite) sequence $S_0; S_1; \ldots; S_n; \ldots$ of states of $\mathcal{A}$, where S_0 is an initial state and every S_{n+1} is obtained from S_n by executing a move of an agent. The partially ordered run, defined in [45], is the most general definition of runs for a distributed ASM. In order to prove properties on a partially ordered run, the attention may be restricted to a linearization of it, which is, in turn, a sequential run (see [45] for more explanations). In the rest of this thesis only *regular runs* in which a state is global and moves of agents are atomic will be considered.

2.3 Digitization Practices

The document [70] "Recommendations for coordination of digitization of cultural heritage in South-Eastern Europe" accepted at the South-Eastern Europe regional meeting on digitization of cultural heritage (Ohrid, Macedonia, 17-20 March 2005) said that the digitization practice in SEE was not matching the priorities communicated on the EU-level and that the rich cultural content of the region was underrepresented in the electronic space. This situation was not much changed during past years. One of the main principles accepted by the participants of the Meeting said that "It is recognized that knowledge of the cultural and scientific heritage is essential for taking decisions concerning its digitization and for interpreting the digitized resources. For this reason, inventorying and cataloging should precede or accompany the digitization of cultural and scientific assets."

Concerning the Meeting conclusions Serbian National Center for Digitization (NCD) recognized the metadata problem as the most sophisticated one in the cataloging phase of digitization. There are a lot of metadata schemes for describing digitized assets of heritage, but not the universal one. Serbian national heritage is described after different standards, according to the nature of assets, but once being digitized, national heritage needs one metadata standard before including in the national database of digital objects. The standard should guarantee interoperability among resources available by different providers and compatibility with the most popular existing international standards.

2.3.1 Situation in Serbia

At the moment, there is no wide spread metadata standard for describing digitized heritage in Serbia. Actually, although the digitization process is entering in the most of the institution caring about national heritage, there is no metadata standard formally accepted at the state level. Different providers of heritage resources (libraries, museums, archives, some research institutions) use international standards appropriate for their specific fields, or ad-hock methods, or old procedures for describing cultural assets in classical format (formulated in 1980s or early 1990s). In fact, some providers wait for some solution of the metadata problem and do not do anything related to digital cataloging. It means that the digital catalogs in Serbia, if exist at all, cannot help in communication between different kinds of providers and users.

2.3.2 International Standards

There are plenty of metadata standards for describing heritage resources, for example:

- Dublin Core [32],
- EAD [36],
- MARC [62],
- TEL AP [38],
- FRBR [49, 83] etc.

Dublin Core is developed and maintained by the Dublin Core Metadata Initiative (DCMI). The goal of DCMI is to promote interoperable metadata standards and develop specialized metadata vocabularies for describing resources. Although this standard can be applied to every kind of resources ensuring interoperability, as it is noted in [16], the problem is that the DC Element Set (Simple DC) is rather restricted and different information must be grouped into one element. More recently, DCMI has allowed some refinements of Simple DC - so called Qualified DC, so that it is possible to develop DC Application Profiles for specific applications and domains. It is important to emphasize that it is possible to lose some information in the process of reducing Qualified DC to Simple DC values. It might be noted that it was possible to use Qualified DC to describe NCD standard, but , in this moment more important issue is to define *what* data should be included in the metadata standard, than to chose *how* to express them in one or the other format.

The other above mentioned standards are mainly related to some special sub-domains of heritage. Encoded Archival Description (EAD) is focused on the archival resources with the goal to allow describing them and to make them accessible to users.

Similarly, the MARC standards are used mostly for the representation and communication of bibliographic and related information in machine-readable form. Another librarian-supported standard is The European Library Application Profile (TEL AP). TEL AP for objects is based on the proposal of Dublin Core Metadata Initiative - Libraries Working Group for the Library Application Profile from 2002. TEL Metadata Working Group is responsible for additions and changes made on the basic document for TEL purposes, the functionality of the Portal at the first place. National Library of Serbia became the full partner of The European Library in 2005 and getting known with the TEL AP concept was a kind of inspiration for the NCD metadata model. The TEL Metadata Registry of terms is currently in development, as well as TEL AP for objects, TEL AP for collections and future TEL AP for services. Functional requirements for bibliographic records (FRBR) is a conceptual, object-oriented model for metadata records, born also in the library world. The model is based on three groups of entities. The first one presents the products of intellectual or artistic endeavor being described, by four entities: work, expression, manifestation and item. Two entities: person and corporate body are in the second group, connected to the responsibility for the intellectual or artistic content, the production and distribution, or the custodianship of these products. The

third group is covering subject area for intellectual or artistic endeavor with four entities: concept, object, event and place. FRBR is extendable to archives, museums and publishing area.

In the field of museums, there is no widely accepted metadata standards. There is a set of recommendations called Spectrum, the UK Museum Documentation Standard [76]. Thus, although these standards are very important and useful in the corresponding sub-domains of digitized heritage, they are partially incompatible, and cannot be directly used to cover the whole domain of heritage.

Having all that in mind, the NCD decided, according to its coordinating role in the digitization field in Serbia, to take care and obligatory to try to solve this metadata problem and define a unique metadata core to assure the interoperability between various digital resources of national cultural and scientific heritage - Proposal of metadata schema for movable cultural heritage.

Digital objects are usually organized in digital collections. A considerable work has gone into building aggregations of digital collections, that was the case with the Digital National Library of Serbia too [28]. Some of the well-known aggregations like Europeana [39] or OAIster [66] have just started to develop collection-level metadata. In 2009, the efforts of NCD metadata working group resulted in a new Proposal of metadata schema for digital collection description. For this schema, The European Library Application Profile for collections [38] was starting point, as well as the MICHAEL collection description [65]. It allows one to create a catalog containing descriptions of physical and/or virtual collections of the already digitized objects.

3

Proof of the Correctness of the Chord Protocol

3.1 Description of Chord Using Abstract State Machines

3.1.1 Basic Notions

Let L, M and K be three positive integer constants such that $K \leqslant 2^M \leqslant L$. Let $N = 2^M$. The following disjoint universes will considered:

- the set $Peer = \{p_1, \ldots, p_L\}$ of all peers that might participate in the considered Chord network,
- the set $Key = \{k_1, \ldots, k_K\}$ of identifiers of objects that might be stored in the considered Chord network, and the set $Value = \{v_1, \ldots, v_K\}$ of the values of those K objects,
- the set $Chord = \{0, 1, \ldots, N-1\}$ denoting at most N peers that are involved in the network in a particular moment,
- the sets $Join = \{join, skip\}$ and $Action = \{put, fair\_leave, unfair\_leave, get, skip\}$ which represent the actions of the peers,
- the sets
  - $Mode = \{not\_connected, connected\}$,
  - $ModeJoin = \{undef, finished, wait\_for\_successor, wait\_for\_keys\}$,
  - $ModeLeave = \{undef, wait\_for\_successor, proceed\_to\_finish\}$,
  - $ModeStabilize = \{undef, wait\_for\_successor, wait\_for\_keys, wait\_for\_predecessor, wait\_for\_successor\_keys, finished\}$,
  - $ModeGet = \{undef, wait\_for\_key, wait\_for\_value\}$,
  - $ModePut = \{undef, wait\_for\_response\}$,
  - $ModeFingers = \{undef, wait\_for\_response\}$

  which represent the states of the peers,

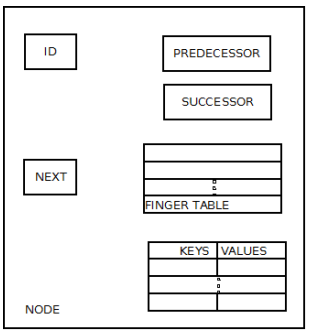


Figure 3.1: Structure of Chord node

- the set $MessageType = \{find\_successor, send\_successor, get, send\_predecessor, set\_predecessor, successor\_found, received\_keyvalue, send\_keys, get\_predecessor, received\_predecessor, value\_found, request\_and\_remove\_keyvalue\}$ of special nullary functions denoting type of communication,
- the set $ContentType = \{Chord \times Value\}^* \cup \{Chord \times Chord\} \cup Chord \cup Value$ which represents all possible types of messages content, and
- the set $Message = MessageType \times ContentType$ which represents messages that are exchanged by the nodes. A messages is defined by its type and content.

Note that:

- it might be that $L > N$ ($K > N$), i.e., that there are more peers (objects to be stored in the network) than nodes, but it can never be $N > L$, and
- without any loss of generality it can be assumed that the numbers of keys and values are the same; if there are more values than keys, all values mapped to the same key might be organized in a list.

In the sequel, the following (standard) list-manipulation functions will be used: *list constructor*, *add* a new element to the list, *remove* an element from the list, and *listitem* returns the i^{th} element of a list; and the strict and the total orders of $\mathbb{N}$ (denoted by $<$, and $\leqslant$). Also, $a \oplus_N b$ will be used to denote $(a+b) \mod N$.

Any peer, active in the network will be called a *node*. A *node* (Figure 3.1) will be represented by its identifier *id* in the network, information on its *predecessor* and *successor*, a *finger table*, a pointer (*next*) to an element in the finger table which will be updated in the current stabilization cycle, and a *list of* $\langle key, value \rangle$ *pairs* of the records for which the node is responsible for.

More formally, the following functions are introduced:

- $id : Peer \to Chord \cup \{undef\}$
- $successor : Chord \to Chord \cup \{undef\}$,
- $predecessor : Chord \to Chord \cup \{undef\}$,
- $finger : Chord \to \{Chord \cup \{undef\}\}^*$,

Function	**Description**
$hash$	Maps the sets of peers and keys to $Chord \cup \{undef\}$
$ping$	Tests whether a node is reachable
$member\_of$	Checks whether a node is between two nodes in $Chord$
$communication$	Realizes communication requests
$mode$	Determines a state of a peer
$mode\_join$	Determines a state of a peer during join operation
$mode\_leave$	Determines a state of a peer during fair leave operation
$mode\_stabilize$	Determines a state of a peer during stabilize operation
$mode\_fingers$	Determines a state of a peer during update of finger table member
$mode\_put$	Determines a state of a peer during put operation
$mode\_get$	Determines a state of a peer during get operation
$known\_nodes$	Simulates external knowledge about existing nodes
key_value	Select a $\langle key, value \rangle$ for storing in the network
$keys$	Select to look for a $value$ with particular key

Table 3.1: Chord functions

- $next : Chord \rightarrow \{1, \ldots, M\}$, and
- $keyvalue : Chord \rightarrow (Chord \times Value)^*$,

where $Chord^*$ is the set that contains lists of the identifiers of nodes, and $(Chord \times Value)^*$ is the set of lists containing pairs $\langle hash(key), value \rangle$. Each $finger(x)$ has M entries ordered respect to the ring ordering.

In other words, a peer p, which is a node, is represented by the tuple

$$\langle id(p), successor(id(p)), predecessor(id(p)), \\ finger(id(p)), next(id(p)), keyvalue(id(p)) \rangle.$$

All the other functions that will be used in the formal description of the protocol, but that are not parts of specification of nodes, are outlined in Table 3.1 and described below. It is assumed that the five functions in Table 3.1 ($hash$, $ping$, $known\_nodes$, key_value and $keys$) are external.

The $hash$ function assigns identifiers of nodes to peers and keys:

- $hash : Peer \cup Key \rightarrow Chord \cup \{undef\}$,

where $undef$ is a special value used in the situations when $hash$ function tries to assign new id for the new:

- peer and there are already N nodes in the network, or
- key and there are already N keys in the network.

The function must also guarantee that in each moment two different active peers (keys) have different hash values. However, note that it is possible that in different moments different peers have the same identifier. Also, it may happen that a peer can have different identifiers (obtained by different calls of the $hash$ function before and after a period in which the peer is not present in

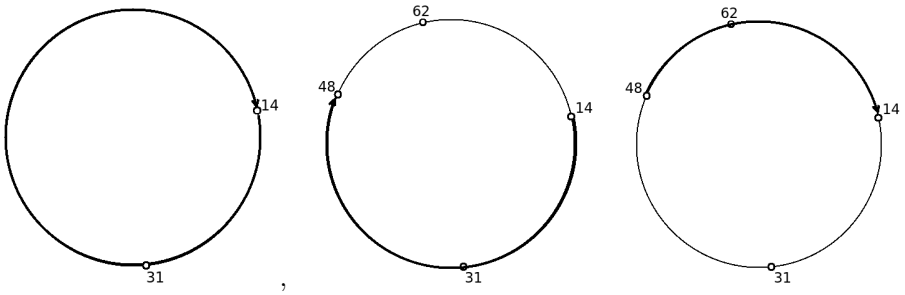


Figure 3.2: Function $member\_of$

the network). The above mentioned *id* function can be explained as a "local" counterpart of *hash*. Namely, it can be assumed that the values produced by *hash* are stored in the local memory and read and published by *id*. In the program given below, *id* will be invoked with the argument Me to allow a node to identify itself in the network.

The external function *ping*, defined as:

- $ping : Chord \to \{true, false\}$

returns *true* or $false$, depending on whether the argument is reachable in the network.

The function $member\_of$:

- $member\_of : Chord \times Chord \times Chord \to \{true, false\}$,

determines whether the first argument is between two next two arguments with respect to the ring ordering, more formally:

- if $arg_2 = arg_3$ always returns *true*,
- if $arg_2 < arg_3$ returns *true* if $arg_2 < arg_1 \leqslant arg_3$ holds,
- if $arg_2 > arg_3$ returns *true* if $\neg(arg_3 < arg_1 \leqslant arg_2)$ holds,
- otherwise returns $false$.

Example 3.1.1. *Let* $N = 64$. *Then,*

- $member\_of(31, 14, 14) =$ *true, see Figure 3.2.a,*
- $member\_of(31, 14, 48) =$ *true, see Figure 3.2.b,*
- $member\_of(62, 14, 48) =$ *false, see Figure 3.2.b,*
- $member\_of(31, 48, 14) =$ *false, see Figure 3.2.c,*
- $member\_of(62, 48, 14) =$ *true, see Figure 3.2.c.*

□

The communication will be realized following the ideas from [8] and [12]. During communication a direct channel is open between two nodes (assuming that channels do not loose information):

- $communication : Chord \times Chord \rightarrow Message^*$.

The first argument is the *sender* and the second one is the *receiver* of a message. When a *sender* sends a new message to a *receiver*, it appends it to the list $Message^*$ of all messages which where sent by this *sender* to that *receiver*. When the *receiver* processes this message, it removes it from the list. Different type of messages contain different information:

- if the type of message is $request\_and\_remove\_keyvalue$ or $set\_predecessor$, these messages will be used just to alert $receiver$, so the content of these messages is empty,
- if the type of message is $send\_keys$, the content of the message is the list $keyvalue(sender)$,
- if the type of message is $get\_predecessor$, the content of the message is $predecessor(sender)$,
- if the type of message is $find\_successor$, the content of the message is $\langle id_1, id_2 \rangle$ where id_1 denotes the value for which the successor is asked, and the id_2 denotes the peer which initiated the query,
- if the type of message is get, the content of the message is $hash(key)$,
- if the type of message is $successor\_found$, the content of the message is $v \in Chord$ for which $member\_of(v, sender, successor(sender)) = true$,
- if the type of message is $value\_found$, the content of the message is the resulting $V \in Value$,
- if the type of message is $received\_keyvalue$, the content of the message is list that will be stored as $keyvalue(receiver)$,
- if the type of message is $received\_predecessor$, the content of the message is id that will be stored as $predecessor(receiver)$.

The function $mode$:

- $mode : Peer \rightarrow Mode$

determines a state of a peer. Initially, for all $p \in Peer$, the value of $mode(p)$ is set to $not\_connected$.

The functions $mode\_join$, $mode\_leave$, $mode\_stabilize$, $mode\_get$, $mode\_put$ and $mode\_fingers$:

- $mode\_join : Peer \rightarrow ModeJoin$
- $mode\_leave : Peer \rightarrow ModeLeave$
- $mode\_stabilize : Peer \rightarrow ModeStabilize$
- $mode\_put : Peer \rightarrow ModePut$
- $mode\_get : Peer \rightarrow ModeGet$
- $mode\_fingers : Peer \rightarrow ModeFingers$

express states of peers as the various Chord operations influence them. Initially, for all $p \in Peer$, the values of these functions are set to *undef*.

The external function *known_nodes*:

- $known\_nodes : Peer \rightarrow Chord$

simulates external knowledge about the nodes in the particular Chord network.

The external function *key_value*:

- $key\_value : Peer \rightarrow Key \times Value$

simulates the choice of a node to store a $\langle key, value \rangle$ pair in the Chord network.

The external function *keys*:

- $keys : Peer \rightarrow Key$

simulates the choice of a node to look if some *value* with particular *key* is stored in the Chord network.

3.1.2 Chord Rules

The rest of this section contains formal description using formalism of ASM of the Chord protocol. The general program is executed by every peer, and a high level description of the rules performed in a Chord network which corresponds to the pseudo code given in [80] (note that the rules FairLeave, UnfairLeave, Put and Get are not given there). A detailed specification of these rules is provided in the Appendix A.

Peer_agent Module

The following main module contains actions that are executed by every peer. The mode of all peers is initially *not_connected*. After a node joins a network successfully, its mode is changed to *connected*. In each execution of a loop, a node concurrently calls the rules responsible for the ring topology maintenance (Stabilize, UpdatePredecessor, UpdateFingers) and communication (ReadMessages) and, according to a non-deterministic choice, it might also invoke one of the FairLeave, UnfairLeave, Put and Get rules.

```
MainModule=
if mode(Me) = not_connected then
  if Choosed Action Is Join
    seq
      if There Are No Known Nodes then
        Start
      else
        Join
      endif
      if Connection Successful then
        mode(Me) := connected
      else
        mode(Me) := not_connected
      endif
    endseq
```

```
  endif
else
  if mode(Me) = connected then
    if id(Me) Does Not Have Communication Problems then
      par
        ReadMessages
        Stabilize
        UpdatePredecessor
        UpdateFingers
        ExtendedJoinModel =
          seq
            choose action in Action
            par
              LeavingActions =
                FairLeave Or UnfairLeave
              KeyValueHandling =
                Put Or Get
            endpar
          endseq
      endpar
    else
      mode(Me) := not_connected
    endif
  endif
endif
```

Chord Rules - High Level Description

In the sequel, the rules of the Chord protocol (listed in Table 3.2) will be described.

When $Node_i$ starts a new *Chord network*, the following rule is executed:

```
Start=
seq
  id(Me) := hash(Me)
  Initialize Values For Node id(Me)
endseq
```

When $Node_i$ asks $Node_j$, which is the member of a *Chord network*, to join the network the following rule is executed (a network is fulfilled if it already contains N nodes):

```
Join=
seq
  id(Me) := hash(Me)
  if Chord Is Not Fulfilled then
    seq
      Initialize Values For Node id(Me) With Respect
        To known
      Take keyvalue That Should Belong To
        Node id(Me) From successor(id(Me))
```

```
    endseq
  endif
endseq
```

After an application of the join rule, all keys which have the hash value less or equal then the *id* of the new node, and which have been stored in the *keyvalue* list of the successor of the new node are moved to the *keyvalue* list of the new node.

When $Node_i$ leaves a *Chord network* in a fair way, it executes the following rule:

```
FairLeave=
seq
  if successor(id(Me)) Is Not Alive then Update successor(id(Me))
  endif
  if id(Me) Is Not The Only Node In Chord then
    seq
      par
        Give keyvalue(id(Me)) To successor(id(Me)))
        Remove id(Me) From Successor Pointers Ring
      endpar
      Deactivate Values For Node id(Me)
    endseq
  endif
endseq
```

Note that a node can also leave a *Chord network* in an unfair way, for example caused by a node crash, communication problems, etc. As it will be shown in Section 3.1.3 the other rules can detect such situations. Thus, the body of UnfairLeave rule is empty.

The next three rules are responsible for detecting situations in which the considered *Chord network* is not in a stable state, updating information about the network and reconnecting the successor pointers of nodes to establish a stable state. These rules are applied periodically, while a peer belongs to the considered *Chord network*:

```
Stabilize=
if successor(id(Me)) Is Alive then
  seq
    Set x To Be predecessor(successor(id(Me)))
    if (x ≠ undef ∧ member_of(x, id(Me), successor(id(Me)))) then
      seq
        successor(id(Me)) := x
        Take Keyvalue That Should Belong To
          Node id(Me) From successor(id(Me))
      endseq
    endif
    if x = undef∨(x ≠ undef∧member_of(id(Me), x, successor(id(Me))))
      then
        Notify successor(id(Me)) To
          Update Its predecessor To id(Me)
    endif
```

Rule	**Description**	**Resulting State**
Start	The first node starts the network	A state with one node
Join	A new node joins the network	A state with an additional node
FairLeave	A node leaves fairly the network	A state without one node
UnfairLeave	A node leaves/crashes	A state without one node
Stabilize	A node updates its successor and predecessor	Successor and predecessor update
UpdatePredecessor	Periodic check of the predecessor	Predecessor update
UpdateFingers	A node runs update on its finger table	Updating finger table entries
Put	A new $\langle key, value \rangle$ pair is stored	Updating $\langle key, value \rangle$ table
Get	Finding a value for a given key	Unchanged state
FindSuccessor	Finding a responsible node for given key or successor of a node	Unchanged state
ReadMessage	Read messages dedicated to a node	Changing some local variables if it is requested

Table 3.2: Chord rules

```
  endseq
else
  Update successor(id(Me))
endif
```

STABILIZE is responsible to update information on successor of a node either because the successor is crashed or the new node with appropriate *id* has enter the network, and to notify new successor about its new predecessor.

Checking if the current predecessor of a node is still active is realized by:

```
UPDATEPREDECESSOR=
if predecessor(id(Me)) Is Not Alive then
  Deactivate Values For predecessor(id(Me))
endif
```

Updating the values from finger table is realized by firing:

```
UPDATEFINGERS=
For Next next(id(Me)) Update finger.listitem(next(id(Me)))
  With Responsible Node For id(Me) ⊕_N 2^(next(id(Me))−1)
```

During one cycle of *Peer_ agent Module* exactly one value from finger table is updated.

Storing a new $\langle key, value\rangle$ pair is realized by the following rule:

```
PUT=
if Chord Is Not Fulfilled then
  Notify Responsible Node For hash(key) To Add
  ⟨hash(key), value⟩ To Its keyvalue Table
endif
```

The name of the rule FINDSUCCESSOR comes from [78]. By this rule a node is asked to return successor of the given argument. The corresponding argument can be the hash value of a key, or the *id* of a peer. In the former case, the result is the identifier of the member of the network which is responsible for the $\langle key, value\rangle$ pair. In the later case, the rule gives the identifier of the first member of the network which is equal to, or greater then, the argument. Note that, if the argument is the *id* of a node (i.e., a peer that is active in a network), the result is *id*, and not the identifier of its successor.

```
FINDSUCCESOR=
For Given key
  if member_of(key, id(Me), successor(id(Me))) then
    Respond With successor(id(Me))
  else
    Forward Query To Closes Predecessor From finger(id(Me))
  endif
```

In the above rule, for the argument h, the current node n returns its successor to the node which was started the query, if $member\_of(h, n, successor(n)) = true$. Otherwise, the query is forwarded to the node whose *id* precedes h in n's finger table, or is the maximal element of this table.

Example 3.1.2. *Let* $N = 64$ *and the nodes with the ids* $\{1, 14, 28, 31, 32, 48\}$ *form a Chord network (see Figure 3.3). Let the corresponding successor pointers*

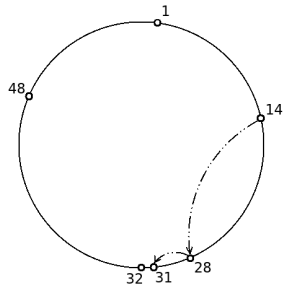


Figure 3.3: Rule FindSuccessor

form a ring. Note that there is no node with the identifier 29 *in the network. The result of call of the successor of* 29 *by the node* 14 *is* 31*, because node* 29 *is understood as the hash of a key or the id of a node entering the network. The node* 14 *will transfer the query to its successor (node* 28*). This node will return its successor (node* 31*) as the result, because* $member\_of(29, 28, 31)$ *is true.*

However, since the node with the identifier 31 *exists in the network, the result of the call of the successor of* 31 *will be* 31*.*

Finally, note that, if one needs information on the successor of a node, for example the node 31*, then it should call for a successor of* $31 \oplus_{64} 1$*.* □

Any node present in a Chord network can execute Get rule (ask for the value of a key). That rule does not change the actual state of the network, but it is defined as:

Get=
Invoke FindSuccessor For Given *key*
And Check Corresponding *value*

During the each execution of a *Peer_ agent Module* all the messages send to a node are processed:

ReadMessages=
Read Messages Dedicated To *Me*,
Change Local Variables If It Is Requested And
Clear Processed Messages

3.1.3 Correctness of the Formalization

In this section the correctness of our ASM formal description of the Chord protocol will be presented with respect to the so-called regular runs. First, following [45] a distributive algebra[1] $\mathcal{A}$ will be formally considered. It consists

[1] [45, Section 6.2]: A distributed algebra $\mathcal{A}$ consists of the following:

- A finite indexed set of single-agent programs π_ν, called *modules*. The *module names* ν are static nullary function names.
- A vocabulary $\Upsilon = Fun(\mathcal{A})$ which includes each $Fun(\pi_\nu) - Self$ but does not contain $Self$. In addition, Υ contains a unary function name Mod.
- A collection of Υ-states, called *initial states* of $\mathcal{A}$, satisfying the following conditions:
  - Different module names are interpreted as different elements.

of the module *Peer_agent* introduced in Section 3.1.2, a vocabulary Υ, and a collection of Υ-states, such that:

- the *vocabulary* Υ contains:
  - all function names that appear in the module $Main$, except Me,
  - a nullary function name *undef*, the standard Boolean constants and operations, and
  - a unary function name Mod,
- a *state* S of the *vocabulary* Υ is a pair which consists of:
  - a base set $Peer \cup Key \cup Value \cup Chord \cup Action \cup Communication \cup \{Main\}$, where $Main$ is a module name, and
  - an Υ-interpretation I which satisfies that

    $$I(Mod) : Peer \to \{Main\},$$

    while the other function names are interpreted as the corresponding objects defined at the beginning of Section 3.1,
- in a local state (of the *vocabulary* $\Upsilon \cup \{Me\}$) which corresponds to the peer $p \in Peer$ and the state S, denoted by *View(p; S)*, all symbols are interpreted as in S, and additionally $I(Me) = p$.

In the initial state a value $mode(p)$ is set to *not_connected* and $id(p)$ is set to *undef* for all $p \in Peer$, while values of $successor(c)$, $predecessor(c)$, $finger(c)$, $next(c)$ and $keyvalue(c)$ are set to *undef* for all $c \in Chord$.

Definition 3.1.1. *Let $x_1, x_2 \in Node$ and $y_0, \ldots y_r \in Node$ be all the nodes from a Chord network such that $y_0 = x_1$, $y_r = x_2$ and $member\_of(y_{i+1}, y_i, y_{i+2}) = true$ for all $i \in \{0, \ldots, r-2\}$. The pair $\langle x_1, x_2 \rangle$ forms a* stable pair *in a state if the following holds:*

- *$y_{i+1} = successor(y_i)$, $y_i = predecessor(y_{i+1})$, for all $i \in \{0, \ldots, r-1\}$.*

A Chord network $\{x_0, \ldots, x_{k-1}\}$, $k \geqslant 1$, is stable *in a state if the pair $\langle x_0, x_0 \rangle$ is stable.* □

Intuitively, a pair $\langle x_1, x_2 \rangle$ is stable in a state if there is no node trying to join the network through the node on the ring-interval (x_1, x_2) in that state and there is not a break in a successors' pointers chain.

A move at a state S is an execution of the module $Main$ by a peer p at *View(p; S)*. The corresponding rules belong to the low-level description given in Appendix A. Similarly as in [45], only atomic moves will be considered.

A run[2] of our distributive algebra $\mathcal{A}$ is a triple $\langle Moves, P, \sigma \rangle$ such that:

- There are only finitely many elements a such that, for some module name ν, $Mod(a) = \nu$.

[2]Note that this is a simplification of the corresponding definition from [45] which results from the fact that the set *Peer* is fixed at the beginning and cannot be changed during executions of Chord.

- $Moves$ is a partially ordered set of moves of peers such that every set $\{y : y \leqslant x\}$ is finite, and $y < x$ means that the move y must be finished before x begins,

- the function P associates with every $x \in Moves$ a peer $P(x)$ which executes x, and satisfies that each nonempty set $\{x : P(x) = p\}$ is linearly ordered, and

- the function σ associates with every finite initial segment of $Moves$ a state of $\mathcal{A}$, while $\sigma(\emptyset)$ denotes an initial state.

Furthermore, it is required that the following coherence condition holds for every run:

- Let x be a maximal element of a finite initial segment X of $Moves$, and $Y = X \setminus \{x\}$. Then, x transforms $\sigma(Y)$ to $\sigma(X)$.

A state is reachable in a run $\langle Moves, P, \sigma\rangle$ if it belongs to the range of σ.

A run $\rho' = \langle Moves', P', \sigma'\rangle$ is an initial segment of a run $\rho = \langle Moves, P, \sigma\rangle$ if $Moves'$ is an initial segment of $Moves$ and P' and σ' are restrictions of P and σ, respectively. A run ρ' is a linearization of a run ρ if $Moves'$ is a linearization[3] of $Moves$, P' and P coincide, and σ' is a restriction of σ. Note that each linearization is a sequential run. Two fundamental corollaries formulated in [45] are:

Corollary 3.1.1. *1. All linearizations of the same finite initial segment of a run have the same final state.*

2. A property holds in every reachable state of a run ρ iff it holds in every reachable state of every linearization of ρ.

Definition 3.1.2. Regular runs *are all runs of a distributive algebra $\mathcal{A}$ which satisfy that:*

- *any execution of* FairLeave, UnfairLeave *and* Put *might happen only between a stable pair of nodes,*

- *the fairness condition, in the sense that every node will eventually execute its next move.* □

The following example illustrates the need for the above constraint. In the example and in the rest of this section the sequences of moves will be graphically illustrated, so that S_i denotes a state, the updated values are in bold, and $\diamondsuit$ means that the rest of a network is not affected by a move.

Example 3.1.3. *Let S_0 be the initial state in which the nodes N_1 and N_3 are members of a network, and the node N_2 wants to join. Suppose that before the pair $\langle N_1, N_3\rangle$ becomes stable, N_1 executes the put rule with the hash 2 of a key. Since N_1 is not aware of N_2, the corresponding key will be stored in N_3, and not in N_2.*

[3] $Moves'$ is linearly ordered, $Moves'$ and $Moves$ have the same elements, and if $x < y$ in $Moves$, then $x < y$ in $Moves'$.

S_0		
id	1	3
predecessor	♢	1
successor	3	♢
hash(key)	*empty*	*empty*

$\xrightarrow{N_2\text{Join}}$

S_1			
id	1	**2**	3
predecessor	♢	***undef***	1
successor	3	**3**	♢
hash(key)	*empty*	***empty***	*empty*

$\xrightarrow{N_2\text{Stabilize}}$

S_2			
id	1	2	3
predecessor	♢	*undef*	**2**
successor	3	3	♢
hash(key)	*empty*	*empty*	*empty*

$\xrightarrow{N_1\text{Put}(Key2)}$

S_3			
id	1	2	3
predecessor	♢	*undef*	2
successor	3	3	♢
hash(key)	*empty*	*empty*	**2**

$\xrightarrow{N_1\text{Stabilize}}$

S_4			
id	1	2	3
predecessor	♢	*undef*	2
successor	**2**	3	♢
hash(key)	*empty*	*empty*	2

$\xrightarrow{N_1\text{Stabilize}}$

S_5			
id	1	2	3
predecessor	♢	**1**	2
successor	2	3	♢
hash(key)	*empty*	*empty*	2

Again, assume that S_0 is the initial state and the network contains the nodes N_1, N_3 and N_4. If the node N_2 executes the join rule, and before the pair $\langle N_1, N_4\rangle$ becomes stable, N_3 wants to leave, N_2 will be isolated from the rest of the network, and the other nodes will never be aware of it.

S_0			
id	1	3	4
predecessor	♢	1	3
successor	3	4	♢

$\xrightarrow{N_2\text{Join}}$

S_1				
id	1	**2**	3	4
predecessor	♢	***undef***	1	3
successor	3	**3**	4	♢

$\xrightarrow{N_3\text{FairLeave}}$

S_2			
id	1	2	4
predecessor	♢	*undef*	1
successor	4	3	♢

A similar example can be given for UnfairLeave. □

In the sequel, the executions of Chord networks will be analyzed. It will be shown that a stable pair of nodes in a Chord network, which executes a regular run, eventually becomes stable after adding/removing of a node between them (the theorems 3.1.1-3.1.6). Corollary 3.1.3 formulates the corresponding statement for a stable network. Finally, it will be proved that the proposed key-handling correctly distributes keys and answers queries (Theorem 3.1.7 and Corollary 3.1.4). Table 3.3 summarizes how our results are related to statements from some other papers.

Theorem 3.1.1 expresses that the rule FindSuccessor will terminate in a finite number of steps. It corresponds to [80, Theorem IV.2]. The statements 3.1.2 – 3.1.6 guarantee that the successor and predecessor pointers for each node will be eventually up to date after a node joins, or unfairly leaves the network.

Theorem 3.1.3 expresses the correctness of the model of executions without failures of nodes and corresponds to [80, Theorem IV.3]. In the statements 3.1.4 – 3.1.6 the model of executions with possible failures is considered. Theorem 3.1.6 is the main statement concerning correctness of maintaining topological structure of Chord networks. Theorem 3.1.7 states that $\langle key, value \rangle$ pairs are properly distributed over the network. Finally, Corollary 3.1.4 shows that if GET returns *undef*, the corresponding $\langle key, value \rangle$ pair is not in the network. In the corresponding proofs some finite initial sequences of runs will be used. STABILIZE and UPDATEPREDECESSOR are executed periodically by all nodes in a network, but the only those applications which change the values of the functions *predecessor* and *successor* will be mentioned. Note that, in each proof some fixed linearization of moves will be considered, but according to Corollary 3.1.1, all linearizations of the corresponding regular run will result in the same final state.

Theorem 3.1.1. *Let $n \in Chord$ be the node which fires the rule* FINDSUCCESSOR *for $h \in \{0, 1, \ldots, N-1\}$. Let m' be the minimal element of Chord such that $h \leqslant m'$. If the pair $\langle n, m' \rangle$ is stable in that state, the node n will get the result after a finite number of moves.*

Proof. Let the node n be asked for the successor of h:

- if h is a member of the ring-interval $(n, successor(n)]$, $successor(n)$ as the result is returned; otherwise
- a node k (such that the ring-interval $(k, h]$ is the smallest subset of the ring-interval $(n, h]$ for every node from the finger table of n) is chosen to answer the query.

Then, FINDSUCCESSOR is invoked by the node k. The number of such calls is limited since there are finitely many nodes between n and h. Furthermore, in each step the ring-interval $(k, h]$ shrinks, and since - by the construction - the first element of the finger table of a node is its successor, the node m will be eventually reached such that $h \in (m, successor(m)]$. Note that in each step a node (an active member of the network) is contacted, and that the result is directly sent to the node n which issued the query. Node n must be active at that moment. It cannot execute (UN)FAIRLEAVE before it gets the answer, since (UN)FAIRLEAVE actions can be executed only when nodes do not wait for answers from earlier fired queries. □

Theorems 3.1.2 – 3.1.6 guarantee that the successor and predecessor pointers for each node will be eventually up to date after a node joins, or unfair leaves the network. In the corresponding proofs some finite initial sequences of runs will be uses. Due to the fact that the STABILIZE and UPDATEPREDECESSOR are applied periodically by all nodes in a network, only those applications which change the values of the functions *predecessor* and *successor* will be mentioned.

Note that, in each proof some fixed linearization of moves will be considered, but according to Corollary 3.1.1, all linearizations of the corresponding regular run will result in the same final state.

Theorem 3.1.3 corresponds to Theorem IV.3 from [78–80].

Theorem 3.1.2. *Let a peer join a Chord network, between two nodes which constitute a stable pair. Then, there is a number $k > 0$ of steps, such that if no other join rule happens in the meantime, the* Stabilize *rule will bring the starting pair to be stable after k steps.*

Proof. Suppose that the network contains only one node N_{i_1} and that N_{i_2} wants to join. The following sequence of moves will be considered:

$\boldsymbol{S_0}$	
id	i_1
predecessor	i_1
successor	i_1

$\xrightarrow{N_{i_2}\,\textsc{Join}}$

$\boldsymbol{S_1}$		
id	i_1	$\boldsymbol{i_2}$
predecessor	i_1	***undef***
successor	i_1	$\boldsymbol{i_1}$

$\xrightarrow{N_{i_2}\,\textsc{Stabilize}}$

$\boldsymbol{S_2}$		
id	i_1	i_2
predecessor	$\boldsymbol{i_2}$	*undef*
successor	i_1	i_1

At this moment the pair is not stable, and the restriction to the regular runs does not allow the nodes to try to leave the network. Also, according to the assumption of the statement, other peers will not execute the Join rule, which consigns to:

$\boldsymbol{S_2}$

$\xrightarrow{N_{i_1}\,\textsc{Stabilize}}$

$\boldsymbol{S_3}$		
id	i_1	i_2
predecessor	i_2	*undef*
successor	$\boldsymbol{i_2}$	i_1

$\xrightarrow{N_{i_1}\,\textsc{Stabilize}}$

$\boldsymbol{S_4}$		
id	i_1	i_2
predecessor	i_2	$\boldsymbol{i_1}$
successor	i_2	i_1

Obviously, a stable pair has been established, again.

Suppose that there are two or more nodes in the network, and that N_{i_k} wants to join. Let $N_{i_{k-1}}$ and $N_{i_{k+1}}$ be the members of the network such that $successor(i_{k-1}) = i_{k+1}$, and $predecessor(i_{k+1}) = i_{k-1}$ (i.e., $\langle N_{i_{k-1}}, N_{i_{k+1}} \rangle$ is a stable pair). Furthermore, let $member\_of(i_k, i_{k-1}, i_{k+1}) = true$. Then, the following sequence of moves will be considered:

$\boldsymbol{S_0}$		
id	i_{k-1}	i_{k+1}
predecessor	$\diamondsuit$	i_{k-1}
successor	i_{k+1}	$\diamondsuit$

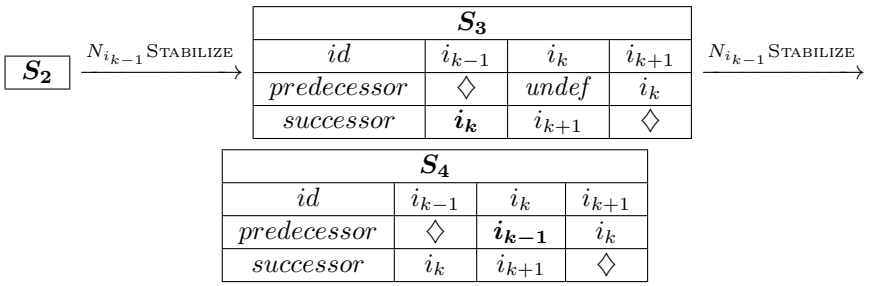
$\xrightarrow{N_{i_k}\,\textsc{Join}}$

$\boldsymbol{S_1}$			
id	i_{k-1}	$\boldsymbol{i_k}$	i_{k+1}
predecessor	$\diamondsuit$	***undef***	i_{k-1}
successor	i_{k+1}	$\boldsymbol{i_{k+1}}$	$\diamondsuit$

$\xrightarrow{N_{i_k}\,\textsc{Stabilize}}$

$\boldsymbol{S_2}$			
id	i_{k-1}	i_k	i_{k+1}
predecessor	$\diamondsuit$	*undef*	$\boldsymbol{i_k}$
successor	i_{k+1}	i_{k+1}	$\diamondsuit$

Similarly as above, the restriction to the regular runs does not allow the nodes to try to leave the network, other peers will not execute the Join rule, and which results with:

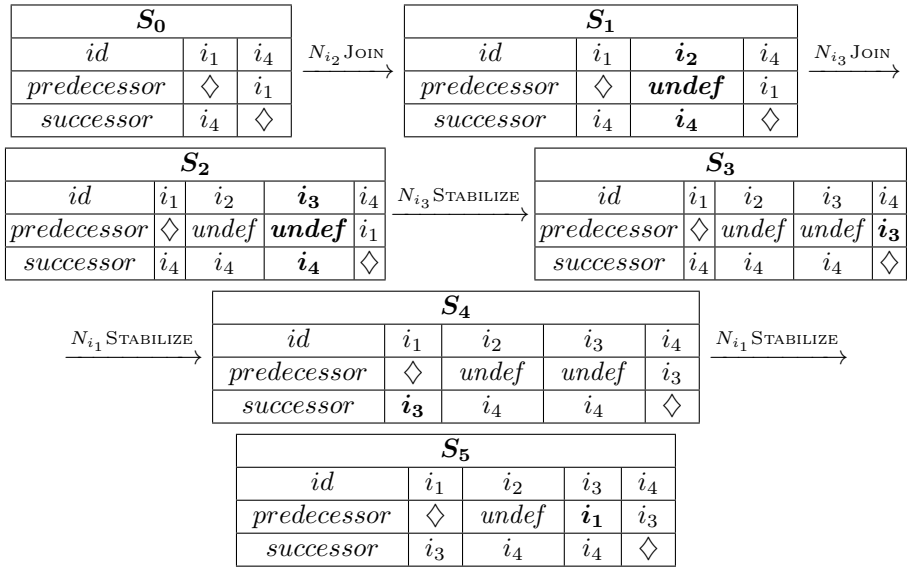


Thus, a stable pair has been established. □

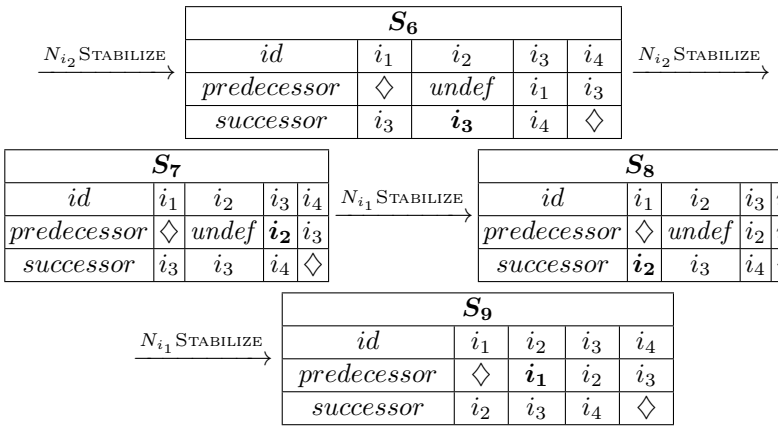
Theorem 3.1.3 (Concurrent joins)**.** *Let a Chord network contain a stable pair. If a sequence of* Join *rules is executed between the nodes which form this stable pair, interleaved with* Stabilize, UpdatePredecessor *and* Update_fingers*, then there is a number $k > 0$ of steps, such that after the last* Join *rule, the starting pair of nodes will be stable after k steps.*

Proof. First, note that UpdateFingers does not change the values of the functions *predecessor* and *successor*. Similarly, UpdatePredecessor might change values of the function *predecessor* only after an UnfairLeave. Thus, executions of UpdatePredecessor and UpdateFingers will not be considered in the rest of this proof.

If it is assumed that all peers that want to join the network have different successors. Then, by Theorem 3.1.2, the statement holds. Otherwise, there must be at least two peers that want to join the network having the same successor. Suppose that N_{i_2} and N_{i_3} want to join and that N_{i_1} and N_{i_4} are members of the network, such that $successor(i_1) = i_4$ and $predecessor(i_4) = i_1$. Furthermore, let $member\_of(i_2, i_1, i_3) = true$ and $member\_of(i_3, i_2, i_4) = true$. Then, the following sequence of moves will be considered:



At this moment, the successor pointers of N_{i_1} and N_{i_3} connect those nodes and N_{i_4}. Then, similarly as in Theorem 3.1.2, executions of Stabilize by N_{i_2} and N_{i_1} result with the stable pair $\langle N_{i_1}, N_{i_4} \rangle$ in the state S_9:



The execution of the Stabilize rule by N_{i_3} in S_2 does not change the values $predecessor(i_2)$ and $successor(i_2)$. The same holds if more than one node (denoted N_{j_1}, N_{j_2}, ...) join the network between N_{i_1} and N_{i_3}. So, if it is assumed that

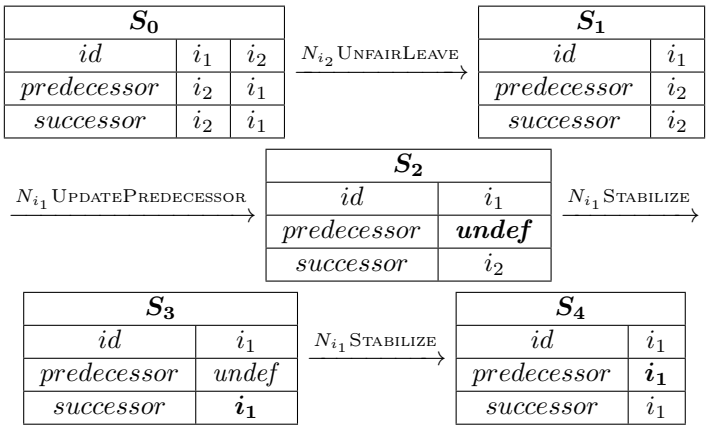
$$i_1 \leqslant \ldots \leqslant j_2 \leqslant j_1 \leqslant i_3 \leqslant i_4,$$

then, similarly as above, a sequence of executions of the Stabilize rule by the nodes N_{j_1}, N_{i_1}, N_{j_2}, N_{i_1}, ... result with a stable starting pair. □

Theorem 3.1.4. *Let a Chord network contain a stable pair and let a node between them leave the network. Then, there is a number $k \geqslant 0$ of steps, such that if no* Join *rule happens at the considered part of the network in the meantime,*

Proof. If it is assumed that the node leaves the network in a fair way, since FairLeave produces a stable pair, the statement holds for $k = 0$. Thus, let UnfairLeave be executed.

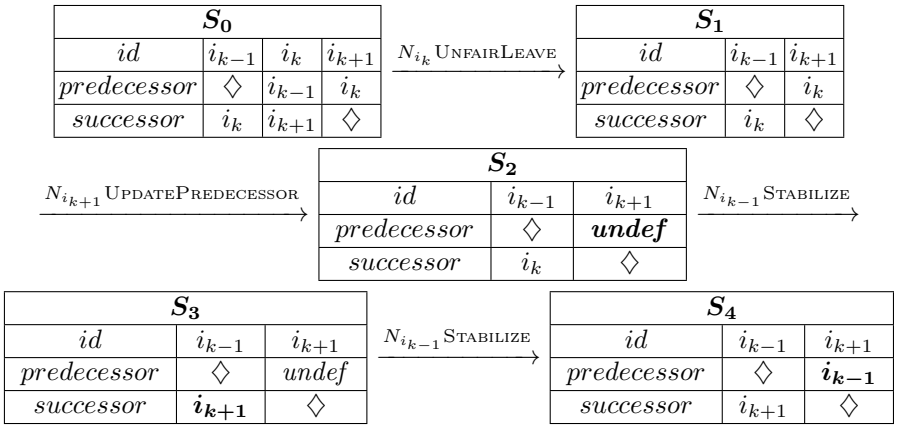
Suppose that the network contains only two nodes N_{i_1} and N_{i_2}, and that N_{i_2} leaves in an unfair way and breaks the ring. Then, the following sequence of moves will be considered:



The state S_4 is stable.

Suppose that there are three or more nodes in a network. Let $N_{i_{k-1}}$, N_{i_k} and $N_{i_{k+1}}$ be the members of the network such that $successor(i_{k-1}) = i_k$ and $successor(i_k) = i_{k+1}$. Suppose that N_{i_k} unfair leaves and breaks the ring of the

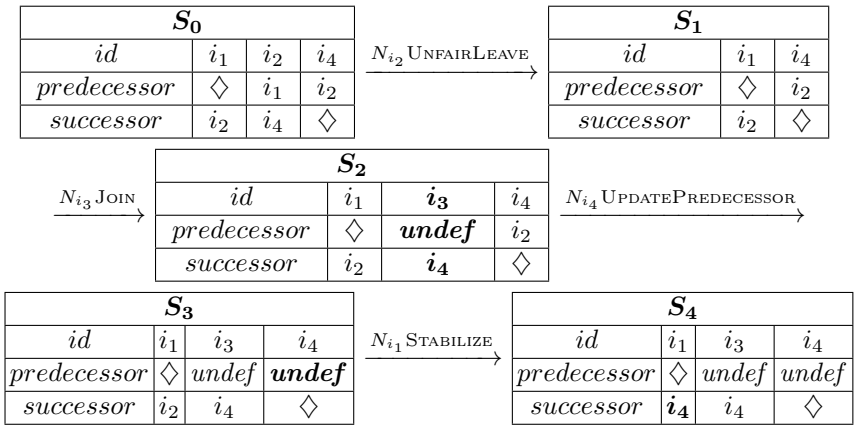
successors pointers. Then, the following sequence of moves which results with the stable pair $\langle N_{i_{k-1}}, N_{i_{k+1}} \rangle$ in state S_4 will be considered:

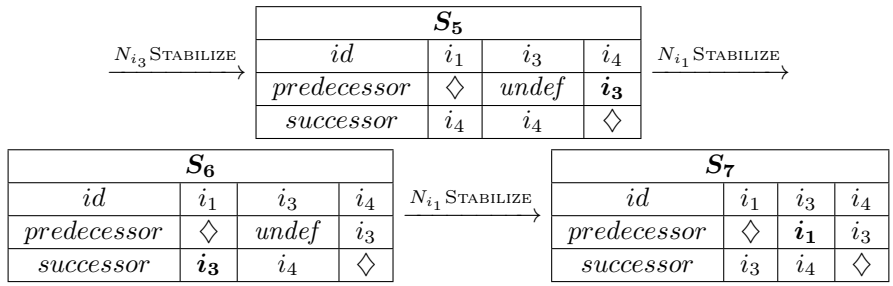


□

Theorem 3.1.5. *Let a Chord network contain a stable pair. Let a node which is between those nodes leave the network following by several nodes which want to join between them. Then, there is a number $k \geqslant 0$ of steps, such that the considered pair will be brought into a stable state after k steps.*

Proof. If it is assumed that the node leaves the network in a fair way. It produces a stable pair, and according to the theorems 3.1.3 and 3.1.4, the statement holds. Then, let the node N_{i_2} execute UnfairLeave and break the ring. If no node joins the network in the ring interval $[predecessor(i_2), successor(i_2)]$, the statement holds similarly as in theorems 3.1.3 and 3.1.4. Finally, assume that a node joins the network in the ring interval $[predecessor(i_2), successor(i_2)]$. Suppose that N_{i_1}, N_{i_2} and N_{i_4} are members of the network, such that $successor(i_1) = i_2$, $predecessor(i_2) = i_1$, $successor(i_2) = i_4$ and $predecessor(i_4) = i_2$. Furthermore, let $member\_of(i_2, i_1, i_3) = true$ and $member\_of(i_3, i_2, i_4) = true$. Let N_{i_2} be the node that will leave, and N_{i_3} node that will join the network. The following sequence of moves will be considered:





which results with the stable starting pair in the state S_7. If more than one node want to join the network in the considered ring interval, the similar arguments as in the proof of Theorem 3.1.3 can be used to establish the statement. □

Note that the restriction from the formulation of Theorem 3.1.5, that no other leave-rules are allowed after the first one, is not essential. According to the definition of regular runs, leave-rules can be executed only between nodes which constitute a stable pair, and we can consider an execution of a sequence of join rules interleaved with leave-rules, and obtain the same result. The above statement will hold for each subsequence which starts with a leave rule followed by several join rules. Thus, the following corollary holds:

Corollary 3.1.2. *Let a Chord network contain a stable pair. Let a node, which is in between those nodes, leave the network. Then, there is a number $k \geqslant 0$, such that the considered pair of nodes will become stable after k moves.*

Theorem 3.1.6 incorporates all previous ideas, and is the main statement concerning correctness of maintaining topological structure of Chord networks.

Theorem 3.1.6. *Let a finite initial segment of a run produce the state S of a Chord network. Then, for every pair of nodes $n, n' \in Chord$, there is a number $k \geqslant 0$, such that $\langle n, n' \rangle$ will become stable after k moves.*

Proof. The case in which the state S is stable is trivial. So, it can be assumed that S is not stable. According to the definition of regular runs, no leave rule might happen before the network becomes stable. Thus, only a sequence of Join rules interleaved with Stabilize, UpdatePredecessor and UpdateFingers can be executed. Since the number of nodes in the network is limited, the number of join rules in the sequence must be finite. Then, similarly as in Theorem 3.1.5, the statement holds. □

Since a network is stable in a state if all pairs of nodes from the network are stable in that state, the following corollary holds:

Corollary 3.1.3. *Let a finite initial segment of a run produce the state S of a Chord network. Then, there is a number $k \geqslant 0$, such that the network will become stable after k moves.*

Finally, the next two statements say that the presented formalization consistently manipulates distributed keys. Theorem 3.1.7 states that $\langle key, value \rangle$ pairs are properly distributed over the network. Informally, it follows from the facts that for every $n \in Chord$, $hash(key) \leqslant n$ for the keys for which n is responsible for, and that all rules that manipulate $\langle key, value \rangle$ pairs invoke FindSuccessor rule.

Theorem 3.1.7 (Golden rule)**.**

$$\forall(\langle key, value\rangle \in Keys \times Values, n \in Chord)(\langle key, value\rangle \in keyvalue(n)$$

$$\Rightarrow member\_of(hash(key), predecessor(n), n).$$

Proof. Obviously, the statement holds for a Chord network containing only one node. According to the definition, a JOIN executed by N_i moves to N_i the corresponding $\langle key, value\rangle$ pairs stored in its successor. Similarly, FAIRLEAVE and STABILIZE executed by N_i transfer all $\langle key, value\rangle$ pairs from N_i to its successor. When a node leaves the network in a unfair way, all $\langle key, value\rangle$ pairs it stored would be lost. Thus the statement remains true. By the definition, for a given $\langle key, value\rangle$ pair, PUT finds a node responsible for the pair which satisfies the statement. Finally, all the other rules do not manipulate $\langle key, value\rangle$ pairs. □

Corollary 3.1.4 follows from the definition of GET, and the theorems 3.1.1 and 3.1.7:

Corollary 3.1.4. *If* GET *returns undef for some* $key \in Keys$*, then there is no* $value \in Values$ *such that* $(key, value)$ *pair is stored in the Chord network.*

Namely, according to Theorem 3.1.7, all $\langle key, value\rangle$ pairs are stored properly, and from Theorem 3.1.1 GET considers only the $\langle key, value\rangle$ pairs which satisfy condition $member\_of(\ hash(key), predecessor(id(N)), id(N))$ and that are stored in the node N.

3.1.4 Discussion and Related Work

One of the basic ideas behind the results presented in Section 3.1.3 is not all possible runs are considered, but only regular runs. Example 3.1.3 shows that without that restriction the ring topology can be damaged which might result in splitting a Chord network in disjoint parts and/or in loosing data.

	Bakhshi, Gurov	Liben-Nowell et al.	Stoica et al.	Tru'o'ng	Zave	**ASM CHORD**
Find Successor Termination	-	-	Theorem IV.2	-	-	**Theorem 3.1.1**
Pure Join Model	π-calculus	Lemma 5.5	Theorem IV.3	-	Section 3.4 (Simulations)	**Theorem 3.1.3**
Model with Failures	-	Theorem 5.9 (wrt HP)	Theorem IV.5 (A Scenario wrt HP)	Chapter 6 (Counter-examples)	Section 4 (Counter-examples)	**Corollary 3.1.3 (wrt RR)**
$\langle key, value\rangle$ Handling	-	-	-	-	-	**Theorem 3.1.7 Corollary 3.1.4 (wrt RR)**

Table 3.3: Schematic Overview of the Related Work

The request that every execution of the main module by a peer is atomic can be relaxed so that only executions of the Chord-rules are required to be atomic.

In that case it is necessary to introduce TTL mechanism. It means that if some rule does not finish in predefined number of module executions, a node repeats execution of that rule from the beginning.

On the other hand, the atomicity assumption is essential and cannot be completely avoided since it eliminates several observed shortages, for instance when a node tries to join the existing network via another node which tries to leave in the same moment. Another counterexample concerns a Chord-network and a node N_i which leaves the network by executing the FairLeave rule. Then, N_i tries to transfer its *keyvalue*-table to $N_{successor(i)}$. However, suppose that $N_{successor(i)}$ leaves the network in the same moment. Obviously, the content of N_i's *keyvalue*-table will be lost.

Several improvements of Chord protocol are proposed to increase robustness:

- in [78] each Chord node can maintain a list containing the node's first r successors, and
- it is usual in software implementations [29] to make multiple copies of all $(key, value)$-pairs (for example, by different hash functions).

Note that those changes might improve performance of a Chord network, but do not affect correctness. Quite simple generalizations of Example 3.1.3 would produce incorrect behavior.

The Chord protocol is introduced in [78–80]. The papers analyze the protocol, its performance and robustness, under the assumption that the nodes and keys are randomly chosen and give several theorems that involve the phrase *with high probability*, for example: "With high probability, the number of nodes that must be contacted to find a successor in a N-node network is $O(\log N)$" [80, Theorem IV.2]. Theorem 3.1.1 corresponds to that statement, but only proves correctness of the rule FindSuccessor. Theorem [80, Theorem IV.2] considers the so-called pure join model of Chord which does not allow (un)fair-leaving, while here the failures of nodes are permitted, but restrict runs to be regular. If the random distribution is assumed, it is easy to see that the same complexity result can be obtained. On the other hand, in the theorems [80, Theorem IV.5] and [80, Theorem IV.6] behavior of FindSuccessor is analyzed when failures of nodes can happen. Then, to prove high probability of correctness it is necessary to involve the aforementioned improvement of Chord which concerns lists of successors of nodes. The only statement in [80] which avoids the mentioned phrase about high probability is Theorem IV.3. It (corresponds to Theorem 3.1.3 and) proves that inconsistent states produced by executing several concurrent Join-rules are transient, i.e., that after the last Join-rule, nodes in a network will form a cycle. That theorem also considers the pure join model. Although only regular runs are considered here, since in Theorem 3.1.3 (un)fair-leaving is not allowed, the theorem holds for all runs. More general sequences of concurrent joining and leaving are considered in [56], where a lower bound of the rate at which nodes need to maintain the system such that it works correctly is given (again) with high probability. In the previous sense, Corollary 3.1.3 corresponds to the main statement [56, Theorem 5.9] of that paper.

It is not quite clear how to compare these two approaches, and there is benefit from both of them. One can argue that the probabilistic approach (providing lower bounds of probabilities) is useful to study robustness of protocols. On the other hand, the approach described in this thesis enables to identify sequences

of actions leading to (un)stable states of Chord networks, which can be useful when one analyzes properties of systems that incorporate Chord and assume its correctness, as it is the case with non-relational database systems.

In [52] the theory of stochastic processes is used to estimate the probability that a Chord network is in a particular state. In [5, 6] Chord's stabilization algorithm is modeled using the π-calculus and it's correctness is established by proving the equivalence of the corresponding specification and implementation. Possible departures of nodes from a network are not examined in this approach. The thesis [82] presents the results of testing of Chord protocol which is based on random simulations. It reports some failures and proposes modifications in the Join and Stabilization rules. For example, an update of the successor of a node is added to the Stabilize-rule to decrease the number of steps needed to obtain a stable state. The specification of Chord presented in Section 3.1.1 includes that modification. In [85] the Alloy formal language is used to prove correctness of the pure join model. The same formalization produces several counterexamples to correctness of Chord ring-maintenance. For the approach presented here, since only the regular runs are considered, those examples do not cause incorrectness. However, this does not imply that in a more general framework some shortages cannot happen.

Finally, all the mentioned papers mainly consider maintaining of topological structure of Chord networks, and do not analyze handling of $\langle key, value\rangle$-pairs.Theorem 3.1.7 and Corollary 3.1.4 show that consistency of data handling is kept in a model of execution which involves failures of nodes.

3.2 Refinement of the Model

A general framework for formal description of systems with so called zero-time transitions, illustrated through Petri nets as state machines and TRIO as assertion language was introduces in [44]. The key novelty in that approach of modeling zero-time transitions was introduction of infinitesimals in the time flow. More precisely, they have adopted and operationalized a natural assumption that transitions of any particular system from one state to the next are not instantaneous but infinitesimal with respect to the execution time of the entire system.

It could be useful to adopt the abstraction of zero-time transitions when the their duration is so short that it can be neglected w.r.t. the whole system evolution in ASM. The main problem arises at the formal level of deduction. At the begging of the transition at time t_1 the system is at the state s_1, and at the end of the transition at time t_2 the system is at the state s_2. If it is assumed that duration of the transition is 0, the consequence of this that $t_1 = t_2$ and at that time the system is both in state s_1 and in state s_2. This exposes to the risk of formal contradictions, especially for the systems where the time is crucial for the inference.

In this section the new semantics will be introduces to prove the Theorems from Section 3.1.3. Also, the language of the introduced temporal logic will be used to formally describe new set of executions ε-regular runs.

With respect to [7] and [44], in this section a discrete linear time temporal propositional logic adequate for modeling zero-time transitions will be presented. Following the concept of non-instantaneous transitions of a system and discrete

linear time model, the goal is to obtain the time flow isomorphic to concatenation of ω copies of ω, i.e. with ω^2 as the model of the time flow.

Arguably, the most intuitive representation of the ordinal ω^2 is the lexicographically ordered $\omega \times \omega$. For the purpose of this thesis, changes of the first coordinate represent different states of a system, while the changes of the second coordinate represent transitions from one state to the next. Hence, it was natural to introduce the following temporal operators:

- $[\omega]$. It represents next state of a system. In the terms of a time flow, it corresponds to the operation $\alpha \mapsto \alpha + \omega$ on ω^2. Semantically, $[\omega]\phi$ is satisfied in the $\langle i, j\rangle$-th time instant iff ϕ is satisfied in the $\langle i + 1, 0\rangle$-th time instant;

- $[1]$. It represents the infinitesimal change of a system within some state. In terms of a time flow, it behaves like the usual next operator: $[1]\phi$ is satisfied in the $\langle i, j\rangle$-th moment iff ϕ is satisfied in the $\langle i, j+1\rangle$-th moment;

- $\mathtt{U}$. It represents the adequate generalization of the until operator from ω to ω^2. Semantically, $\phi \,\mathtt{U}\, \psi$ is satisfied in the $\langle i, j\rangle$-th moment iff there is $\langle k, l\rangle \geqslant_{\text{lex}} \langle i, j\rangle$ so that ψ is satisfied in the $\langle k, l\rangle$-th moment, and for all $\langle r, s\rangle$ such that $\langle i, j\rangle \leqslant_{\text{lex}} \langle r, s\rangle <_{\text{lex}} \langle k, l\rangle$, ϕ is satisfied in $\langle r, s\rangle$-th moment. Here $\leqslant_{\text{lex}}$ denotes lexicographical ordering;

- $\mathtt{u}$. It is a local version of the until operator. Semantically, $\phi \,\mathtt{u}\, \psi$ is satisfied in the $\langle i, j\rangle$-th moment iff there is a nonnegative integer k such that ψ is satisfied in the $\langle i, j + k\rangle$-th moment, and for all $l < k$, ϕ is satisfied in $\langle i, j + l\rangle$-th moment.

The main technical results are the proofs of the completeness theorem and determination of the complexity for the satisfiability procedure (PSPACE).

3.2.1 Syntax and semantics

Let $Var = \{p_n \mid n \in \omega\}$ be the set of propositional variables. The set For of all $L([1], [\omega], \mathtt{u}, \mathtt{U})$-formulas is the smallest superset of Var that is closed under the following formation rules:

- $\phi \mapsto *\phi$, where $* \in \{\neg, [1], [\omega]\}$;

- $\langle \phi, \psi\rangle \mapsto (\phi * \psi)$, where $* \in \{\wedge, \mathtt{u}, \mathtt{U}\}$.

As it is usual, in order to simplify notation the standard convention of omission of parentheses and the standard priority of connectives will be used (all unary connectives have the greater priority than any binary connective; connectives of the same arity have identical priority). From now on, a formula will stand for an $L([1], [\omega], \mathtt{u}, \mathtt{U})$-formula. Formulas will be denoted by ϕ, ψ and θ, indexed if necessary. The remaining logical and temporal connectives $\vee$, $\rightarrow$, $\leftrightarrow$, $\mathtt{f}$, $\mathtt{g}$, $\mathtt{F}$ and $\mathtt{G}$ are defined in the usual way:

- $\phi \vee \psi =_{\text{def}} \neg(\neg\phi \wedge \neg\psi)$;

- $\phi \rightarrow \psi =_{\text{def}} \neg\phi \vee \psi$;

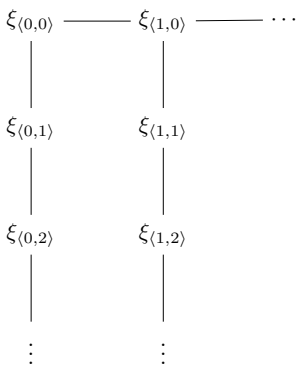


Figure 3.4: Kripke model

- $\phi \leftrightarrow \psi =_{\text{def}} (\phi \to \psi) \wedge (\psi \to \phi)$;
- $\mathtt{f}\phi =_{\text{def}} (\phi \to \phi)\mathtt{u}\phi$;
- $\mathtt{g}\phi =_{\text{def}} \neg\mathtt{f}\neg\phi$;
- $\mathtt{F}\phi =_{\text{def}} (\phi \to \phi)\mathtt{U}\phi$;
- $\mathtt{G}\phi =_{\text{def}} \neg\mathtt{F}\neg\phi$;
- $[a]^0\phi =_{\text{def}} \phi$; $[a]^{n+1}\phi =_{\text{def}} [a][a]^n\phi$, $a \in \{1, \omega\}$.

Nonempty sets of formulas will be called theories.

The time flow is isomorphic to the ordinal $\omega \cdot \omega$ (recall that in ordinal arithmetics $\omega \cdot \omega$ is a concatenation of ω copies of ω). Instead of $\langle \omega \cdot \omega, \in\rangle$, as a canonical ordering the lexicographically ordered $\omega \times \omega$ will be used. Recall that $\langle i, j\rangle \leqslant_{\text{lex}} \langle k, l\rangle$ iff $i < k$, or $i = k$ and $j \leqslant l$. If the context is clear, "lex" will be omitted from the subscript. From now on, the elements of $\omega \times \omega$ would be referred as time instants, and will be denoted by $\mathbf{r}$, $\mathbf{s}$ and $\mathbf{t}$, indexed or primed if necessary. In particular, $\mathbf{r}$ should be regarded as $\langle r_1, r_2\rangle$, $\mathbf{s}$ should be regarded as $\langle s_1, s_2\rangle$ and so on.

Definition 3.2.1. *A model is any function (evaluation) of the form* $\xi : \omega \times \omega \times Var \longrightarrow \{0, 1\}$. □

Models will be denoted by ξ, η and ζ, indexed if necessary. Note that the equivalent Kripke style semantics can be obtained by introducing worlds as valuations of the form $\xi_{i,j}(p) = \xi(i, j, p)$, for every propositional variable p. Thus, the models presented here can be graphically represented as in Figure 3.4.

Definition 3.2.2. *Let* $\xi : \omega \times \omega \times Var \longrightarrow \{0, 1\}$. *The predicate* $\xi \models_{\mathbf{r}} \phi$ *is defined, which reads "a model* ξ *satisfies formula* ϕ *in the* $\mathbf{r}$*-th moment (or in the* $\mathbf{r}$*-th time instant)", recursively on the complexity of formulas as follows:*

1. $\xi \models_{\mathbf{r}} p_n \Leftrightarrow_{\text{def}} \xi(\mathbf{r}, p_n) = 1$;
2. $\xi \models_{\mathbf{r}} \neg\phi \Leftrightarrow_{\text{def}} \xi \not\models_{\mathbf{r}} \phi$;

3. $\xi \models_{\mathbf{r}} \phi \wedge \psi \Leftrightarrow_{\text{def}} \xi \models_{\mathbf{r}} \phi$ *and* $\xi \models_{\mathbf{r}} \psi$;

4. $\xi \models_{\mathbf{r}} [1]\phi \Leftrightarrow_{\text{def}} \xi \models_{\langle r_1, r_2+1\rangle} \phi$;

5. $\xi \models_{\mathbf{r}} [\omega]\phi \Leftrightarrow_{\text{def}} \xi \models_{\langle r_1+1, 0\rangle} \phi$;

6. $\xi \models_{\mathbf{r}} \phi \,\mathtt{u}\, \psi \Leftrightarrow_{\text{def}}$ *there exists* $k \in \omega$ *such that* $\xi \models_{\langle r_1, r_2+k\rangle} \psi$ *and* $\xi \models_{\langle r_1, r_2+i\rangle} \phi$ *for all* $0 \leqslant i < k$;

7. $\xi \models_{\mathbf{r}} \phi \,\mathtt{U}\, \psi \Leftrightarrow_{\text{def}}$ *there exists* $\mathbf{s} \geqslant \mathbf{r}$ *such that* $\xi \models_{\mathbf{s}} \psi$ *and* $\xi \models_{\mathbf{t}} \phi$ *for all* $\mathbf{r} \leqslant \mathbf{t} < \mathbf{s}$. □

A formula ϕ is satisfiable iff there exist a model ξ and a time-instant $\mathbf{t}$ so that $\xi \models_{\mathbf{t}} \phi$. A formula ϕ is valid iff $\xi \models_{\mathbf{t}} \phi$ for all ξ and all $\mathbf{t}$. For instance, an immediate consequence of (2) and (3) of Definition 3.2.2 is validity of any substitutional instance of any classical tautology. A slightly less trivial example of a valid formula is $[1][\omega]\phi \leftrightarrow [\omega]\phi$. Indeed,

$$\begin{aligned} \xi \models_{\mathbf{t}} [1][\omega]\phi &\Leftrightarrow \xi \models_{\langle t_1, t_2+1\rangle} [\omega]\phi \\ &\Leftrightarrow \xi \models_{\langle t_1+1, 0\rangle} \phi \\ &\Leftrightarrow \xi \models_{\mathbf{t}} [\omega]\phi. \end{aligned}$$

Concerning the relationship between the operators $\mathtt{U}$ and $\mathtt{u}$, it is not hard to see that

$$(\phi \,\mathtt{u}\, \psi) \rightarrow (\phi \,\mathtt{U}\, \psi)$$

is valid, but the converse implication is not.

If T is a theory, then $\xi \models_{\mathbf{t}} T$ means that $\xi \models_{\mathbf{t}} \phi$ for all $\phi \in T$, while $T \models \phi$ means that, for all ξ and all $\mathbf{t}$, $\xi \models_{\mathbf{t}} T$ implies $\xi \models_{\mathbf{t}} \phi$. A theory T is satisfiable iff there exist a model ξ and a time-instant $\mathbf{t}$ so that $\xi \models_{\mathbf{t}} T$. A theory T is finitely satisfiable iff all finite subsets of T are satisfiable.

Theorem 3.2.1. *The compactness theorem fails for* $L([1], [\omega], \mathtt{u}, \mathtt{U})$.

Proof. Let

$$T = \{\mathtt{F}\neg p_0\} \cup \{[\omega]^m[1]^n p_0 \mid m, n \in \omega\}.$$

If $\xi \models_{\mathbf{t}} \mathtt{F}\neg p_0$, then there exist $m, n \in \omega$ so that $\xi \models_{\langle t_1+m, t_2+n\rangle} \neg p_0$. As a consequence, $\xi \not\models_{\mathbf{t}} [\omega]^m[1]^n p_0$, so T is unsatisfiable. It remains to show that T is finitely satisfiable.

Let S be a nonempty finite subset of T. Since S is finite, there exists a positive integer k such that $k > \max(m, n)$ for all formulas of the form $[\omega]^m[1]^n p_0$ that appear in S. Let ξ be any model such that $\xi(\mathbf{r}, p_0) = 1$ for all $\mathbf{r} < \langle k, 0\rangle$ and $\xi(k, 0, p_0) = 0$. Then, $\xi \models_{\langle 0,0\rangle} S, \mathtt{F}\neg p_0$, so S is satisfiable. □

3.2.2 Complete axiomatization

Since [1] and $[\omega]$ are essentially modal operators with some additional properties (self-duality for example), one natural way to approach the construction of a syntactical counterpart $\vdash$ of the satisfiability relation $\models$ is to determine sufficient but reasonable amount of properties that enables the construction of the standard monster model (the set of worlds is the set of all saturated theories).

In particular, one must provide that $T \vdash \phi$ implies $[a]T = \{[a]\psi \mid \psi \in T\} \vdash [a]\phi$, $a \in \{1, \omega\}$ and validity of deduction theorem ($T \vdash \phi \to \psi$ iff $T, \phi \vdash \psi$).

However, the situation that is consider here is significantly simpler than the general one. Firstly, all the models have exactly the same frame ω^2. Secondly, satisfiability of propositional letters in any node can be defined syntactically by formulas of the form $[\omega]^m[1]^n p$. Hence, if T is any complete theory, then it is quite reasonable to expect that the function $\xi_T : \omega \times \omega \times Var \longrightarrow \{0, 1\}$ defined by

$$\xi_T(m, n, p) = 1 \text{ iff } T \vdash [\omega]^m[1]^n p$$

is a model of T in a sense that $\xi_T \models_{\langle m,n\rangle} \phi$ iff $T \vdash [\omega]^m[1]^n\phi$ (in particular, $\xi_T \models_{\langle 0,0\rangle} T$).

So, the definition of $\vdash$ will incorporate sufficient amount of semantical properties of $\models$ to ensure formal proof of the fact that ξ_T described above is a model of a complete theory T.

Axioms and inference rules

The axioms of $L([1], [\omega], \mathtt{u}, \mathtt{U})$ are all instances of the following eight schemata:

A1 Tautology instances;

A2 $[1][\omega]\phi \leftrightarrow [\omega]\phi$;

A3 $\neg[a]\phi \leftrightarrow [a]\neg\phi$, $a \in \{1, \omega\}$;

A4 $[a](\phi * \psi) \leftrightarrow ([a]\phi * [a]\psi)$, $[a] \in \{1, \omega\}$ and $* \in \{\wedge, \vee, \to, \leftrightarrow\}$;

A5 $\psi \to (\phi \,\mathtt{u}\, \psi)$;

A6 $\phi \,\mathtt{u}\, \psi \to \phi \,\mathtt{U}\, \psi$;

A7 $\left(\bigwedge_{k=0}^{n}[1]^k(\phi \wedge \neg\psi) \wedge [1]^{n+1}\psi\right) \to \phi \,\mathtt{u}\, \psi$;

A8 $\left(\bigwedge_{k=0}^{n}[\omega]^k\mathtt{g}(\phi \wedge \neg\psi) \wedge [\omega]^{n+1}\phi \,\mathtt{u}\, \psi\right) \to \phi \,\mathtt{U}\, \psi$.

A1 reflects the fact that all tautology instances are valid. A2 captures the interplay between [1] and $[\omega]$, which in ordinal arithmetics can be stated as $1 + \omega = \omega$. A3 and A4 reflect self-duality of both [1] and $[\omega]$. A5, A6, A7 and A8 capture the $\Rightarrow$ part in the following characterization of $\mathtt{u}$ and $\mathtt{U}$:

- $\neg(\phi \,\mathtt{u}\, \psi) \Leftrightarrow \neg\psi \wedge \bigwedge_{n\in\omega}\bigvee_{k=0}^{n}[1]^k(\neg\phi \vee \psi) \vee [1]^{n+1}\neg\psi$;
- $\neg(\phi \,\mathtt{U}\, \psi) \Leftrightarrow \neg(\phi \,\mathtt{u}\, \psi) \wedge \bigwedge_{n\in\omega}\bigvee_{k=0}^{n}[\omega]^k\neg\mathtt{g}(\phi \wedge \neg\psi) \vee [\omega]^{n+1}\neg(\phi \,\mathtt{u}\, \psi)$.

The inference rules of $L([1], [\omega], \mathtt{u}, \mathtt{U})$ are the following four rules:

R1 Modus ponens: from ϕ and $\phi \to \psi$ infer ψ;

R2 Necessitation: from ϕ infer $[a]\phi$, $a \in \{1, \omega\}$;

R3 $\mathtt{u}$-rule: from the set of premises

$$\{\theta \to \neg\psi\} \cup \left\{\theta \to \bigvee_{k=0}^{n}[1]^k(\neg\phi \vee \psi) \vee [1]^{n+1}\neg\psi \mid n \in \omega\right\}$$

infer $\theta \to \neg(\phi \,\mathtt{u}\, \psi)$;

R4 U-rule: from the set of premises

$$\{\theta \to \neg(\phi \,\mathtt{u}\, \psi)\} \cup \left\{ \theta \to \bigvee_{k=0}^{n} [\omega]^k \neg \mathtt{g}(\phi \wedge \neg\psi) \vee [\omega]^{n+1} \neg(\phi \,\mathtt{u}\, \psi) \mid n \in \omega \right\}$$

infer $\theta \to \neg(\phi \,\mathtt{U}\, \psi)$.

Modus ponens and necessitation have the same meaning as in any modal logic with multiple modal operators, see for instance [77]. Rules R3 and R4 reflect the $\Leftarrow$ part of the characterization of u and U.

Basic proof theory

The inference relation $\vdash$ is defined as follows:

Definition 3.2.3. *An $L([1],[\omega],\mathtt{u},\mathtt{U})$-formula ϕ is a* theorem*, denoted by $\vdash \phi$, iff there exists an at most countably infinite sequence of $L([1],[\omega],\mathtt{u},\mathtt{U})$-formulas $\phi_0,\ldots,\phi_\alpha$ (the ordering type of $\phi_0,\ldots,\phi_\alpha$ is $\alpha+1$, where α is any countable ordinal) such that $\phi_\alpha = \phi$ and for all $\beta \leqslant \alpha$, ϕ_β is instance of some axiom, or ϕ_β can be obtained from some previous members of the sequence by application of some inference rule.* □

Definition 3.2.4. *Let T be an $L([1],[\omega],\mathtt{u},\mathtt{U})$-theory and ϕ an $L([1],[\omega],\mathtt{u},\mathtt{U})$-formula. A formula ϕ is syntactical consequence of T (or that ϕ is deducible or derivable from T), denoted by $T \vdash \phi$, iff there exists an at most countably infinite sequence of $L([1],[\omega],\mathtt{u},\mathtt{U})$-formulas $\phi_0,\ldots,\phi_\alpha$ such that $\phi_\alpha = \phi$ and for all $\beta \leqslant \alpha$, ϕ_β is instance of some axiom, $\phi_\beta \in T$, or ϕ_β can be obtained from some previous members of the sequence by application of some inference rule, where the application of necessitation is restricted to theorems.* □

An immediate consequence of the previous two definitions is the fact that structural rules cut ($T \vdash \Gamma$, $\Gamma \vdash \phi$ implies $T \vdash \phi$) and weakening ($T \vdash \phi$ implies $T, \psi \vdash \phi$) are true for the introduced consequence relation $\vdash$. The soundness theorem ($T \vdash \phi$ implies $T \models \phi$) can be straightforwardly proved by the induction on the length of the inference. Let us prove deduction theorem for $L([1],[\omega],\mathtt{u},\mathtt{U})$.

Theorem 3.2.2 (Deduction theorem)**.** *$T \vdash \phi \to \psi$ iff $T, \phi \vdash \psi$.*

Proof. The $\Rightarrow$ part is a straightforward consequence of weakening and modus ponens. The converse implication will be proved by induction on the length of the inference.

If ψ is an axiom instance or $\psi \in T$, then $T \vdash \psi$, so since $T \vdash \psi \to (\phi \to \psi)$ (A1), by modus ponens $T \vdash \phi \to \psi$. If $\psi = \phi$, then by A1, $T \vdash \phi \to \phi$.

Suppose that ψ is a theorem. Then, $\vdash [a]\psi$, so by weakening $T \vdash [a]\psi$, $T \vdash \phi \to [a]\psi$ stands. Thus, it is verified that the $\Leftarrow$ part is preserved under the application of necessitation.

Suppose that $T \vdash \phi \to (\theta \to \neg\psi_2)$ and $T \vdash \phi \to (\theta \to \bigvee_{k=0}^{n}[1]^k(\neg\psi_1 \vee \psi_2) \vee [1]^{n+1}\neg\psi_2)$ for all $n \in \omega$. Since $p \to (q \to r)$ is equivalent to $p \wedge q \to r$, the following holds $T \vdash \phi \wedge \theta \to \neg\psi_2$ and $T \vdash \phi \wedge \theta \to \bigvee_{k=0}^{n}[1]^k(\neg\psi_1 \vee \psi_2) \vee [1]^{n+1}\neg\psi_2)$. By u-rule, $T \vdash \phi \wedge \theta \to \neg(\psi_1 \,\mathtt{u}\, \psi_2)$, i.e. $T \vdash \phi \to (\theta \to \neg(\psi_1 \,\mathtt{u}\, \psi_2))$. Hence, it is verified that the $\Leftarrow$ part is preserved under the application of the u-rule. The remaining step (application of the U-rule) can be proved similarly. □

Lindenbaum's theorem

Roughly speaking, the standard maximization argument in the proof of Lindenbaum's theorem goes as follows: the starting point is a consistent theory and in consecutive steps and it is extended with a single formula or with its negation (depending which choice preserves consistency), until all formulas have been exhausted. Due to the presence of infinitary inference rules, it is necessary to additionally check in each step whether the current formula that is incompatible with the current theory can be derived by application of R3 or R4. If that is the case, at least one premise have to be blocked. Detailed construction is given below.

Lemma 3.2.1. *Suppose that T is a consistent theory and that $T \vdash \neg(\theta \to \neg(\phi \,\mathtt{u}\, \psi))$. Then, $T, \neg(\theta \to \neg\psi)$ is consistent, or there exists a nonnegative integer m such that $T, \neg\left(\theta \to \bigvee_{k=0}^{m}[1]^k(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi\right)$ is consistent.*

Proof. If $T, \neg(\theta \to \neg\psi)$ and $T, \neg\left(\theta \to \bigvee_{k=0}^{m}[1]^k(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi\right)$ are inconsistent for all $m \in \omega$, then $T \vdash \theta \to \neg\psi$ and $T \vdash \theta \to \bigvee_{k=0}^{m}[1]^k(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi$ for all $m \in \omega$, so by R3, $T \vdash \theta \to \neg(\phi \,\mathtt{u}\, \psi)$, which contradicts the fact that T is consistent. □

Lemma 3.2.2. *Suppose that T is a consistent theory and that $T \vdash \neg(\theta \to \neg(\phi \,\mathtt{U}\, \psi))$. Then, $T, \neg(\theta \to \neg(\phi \,\mathtt{u}\, \psi))$ is consistent, or there exists a nonnegative integer m such that $T, \neg\left(\theta \to \bigvee_{k=0}^{m}[\omega]^k\neg\mathtt{g}(\phi \wedge \neg\psi) \vee [\omega]^{m+1}\neg(\phi \,\mathtt{u}\, \psi)\right)$ is consistent.*

Proof. Similar as the proof of Lemma 3.2.1, with the use of R4 instead of R3. □

Theorem 3.2.3 (Lindenbaum's theorem)**.** *Every consistent theory can be maximized, i.e. extended to a complete theory.*

Proof. Let T be a consistent theory and let $\langle \phi_n \mid n \in \omega \rangle$ be one enumeration of the set of all $L([1], [\omega], \mathtt{u}, \mathtt{U})$-formulas. The sequence $\langle T_n \mid n \in \omega \rangle$ of theories will be inductively defined as follows:

1. $T_0 = T$;
2. If T_n is compatible with ϕ_n ($T_n \cup \{\phi_n\}$ is consistent), then let $T_{n+1} = T_n \cup \{\phi_n\}$;
3. If T_n is incompatible with ϕ_n ($T_n \cup \{\phi_n\}$ is inconsistent) and $\phi_n \neq \theta \to \neg(\psi_1 \,\mathtt{u}\, \psi_2), \theta \to \neg(\psi_1 \,\mathtt{U}\, \psi_2)$, then let $T_{n+1} = T_n \cup \{\neg\phi_n\}$;
4. If T_n is incompatible with ϕ_n and $\phi_n = \theta \to \neg(\psi_1 \,\mathtt{u}\, \psi_2)$, then let $T_{n+1} = T_n \cup \{\neg\phi_n, \neg(\theta \to \bigvee_{k=0}^{m}[1]^k(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi)\}$, where m is the smallest nonnegative integer such that T_{n+1} is consistent. If there is no such m, then by Lemma 3.2.1, $T_n \cup \{\neg\phi_n, \neg(\theta \to \neg\psi)\}$ is consistent, so let $T_{n+1} = T_n \cup \{\neg\phi_n, \neg(\theta \to \neg\psi)\}$;
5. If T_n is incompatible with ϕ_n and $\phi_n = \theta \to \neg(\psi_1 \,\mathtt{U}\, \psi_2)$, then let $T_{n+1} = T_n \cup \{\neg\phi_n, \neg(\theta \to \bigvee_{k=0}^{m}[\omega]^k\neg\mathtt{g}(\phi \wedge \neg\psi) \vee [\omega]^{m+1}\neg(\phi \,\mathtt{u}\, \psi))\}$, where m is the least nonnegative integer such that T_{n+1} is consistent. If there is no such m, then by Lemma 3.2.2, $T_n \cup \{\neg\phi_n, \neg(\theta \to \neg(\phi \,\mathtt{u}\, \psi))\}$ is consistent, so let $T_{n+1} = T_n \cup \{\neg\phi_n, \neg(\theta \to \neg(\phi \,\mathtt{u}\, \psi))\}$.

Note that each T_n is consistent. Let $T_\omega = \bigcup_{n\in\omega} T_n$. Clearly, $T_\omega \vdash \phi$ or $T_\omega \vdash \neg\phi$ for any formula ϕ. It remains to show consistency of T_ω. In order to do so, it is sufficient to show that T_ω is deductively closed. Case (2) ensures that all axiom instances and all theorems are in T_ω. Since necessitation can be applied only on theorems, T_ω is closed under its application. It remains to show that T_ω is closed under modus ponens, $\mathtt{u}$-rule and $\mathtt{U}$-rule.

MP: Let $\phi, \phi \to \psi \in T_\omega$. Then, there exist $k, m \in \omega$ so that $\phi \in T_k$ and $\phi \to \psi \in T_m$. Let $\psi = \phi_n$ and let $l = k + m + n + 1$. By construction of T_ω, $T_k, T_m, T_{n+1} \subseteq T_l$, so by modus ponens, $T_l \vdash \psi$. Since T_l is consistent, $\psi = \phi_n$ and $T_n \subseteq T_l$, T_n and ψ are compatible, so $T_{n+1} = T_n \cup \psi$, i.e. $\psi \in T_\omega$.

$\mathtt{u}$-rule: Let $\{\theta \to \neg\psi\} \cup \{\theta \to \bigvee_{i=0}^{m}[1]^i(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi \mid m \in \omega\} \subseteq T_\omega$ and let $\theta \to \neg(\phi\,\mathtt{u}\,\psi) = \phi_k$. Suppose that $\theta \to \neg(\phi\,\mathtt{u}\,\psi) \notin T_\omega$. Then, $T_{k+1} = T_k, \neg\phi_k, \neg(\theta \to \neg\psi)$ or $T_{k+1} = T_k, \neg\phi_k, \neg(\theta \to \bigvee_{i=0}^{m}[1]^i(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi)$. Now, for sufficiently large index n, T_n contains both θ_1 and $\neg\theta_1$, where $\theta_1 = \theta \to \neg\psi$ or $\theta_1 = \theta \to \bigvee_{i=0}^{m}[1]^i(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi$, which contradicts consistency of T_n.

$\mathtt{U}$-rule: Similarly as the previous case. □

Completeness theorem

Due to the fact that the validity of Lindenbaum's theorem for $L([1], [\omega], \mathtt{u}, \mathtt{U})$ has been established, it remains to show that each complete theory T is satisfiable. Thus, trough the rest of this section T will be a complete theory.

Definition 3.2.5. *Let T be a complete theory. The canonical model ξ_T is defined by*

$$\xi_T(k, l, p) = 1 \;\Leftrightarrow_{\text{def}}\; T \vdash [\omega]^k[1]^l p.$$

□

The proof of the fact that $\xi_T \models_{\langle 0,0\rangle} T$ will be devided into the following five lemmas:

Lemma 3.2.3. *Let $T \vdash \phi\,\mathtt{u}\,\psi$. Then, $T \vdash \psi$ or there exists a nonnegative integer m such that $T \vdash \bigwedge_{i=0}^{m}[1]^i(\phi \wedge \neg\psi) \wedge [1]^{m+1}\psi$.*

Proof. Contrary to the assumption of the lemma, let $T \nvdash \psi$ and $T \nvdash \bigwedge_{i=0}^{m}[1]^i(\phi \wedge \neg\psi) \wedge [1]^{m+1}\psi$ for all $m \in \omega$. Since T is complete, $T \vdash \neg\psi$ and $T \vdash \bigvee_{i=0}^{m}[1]^i(\neg\phi \vee \psi) \vee [1]^{m+1}\neg\psi$ for all $m \in \omega$, so by R3 (for θ any axiom instance can be chosen), $T \vdash \neg(\phi\,\mathtt{u}\,\psi)$, which contradicts consistency of T. □

Lemma 3.2.4. *Let $T \vdash \phi\,\mathtt{U}\,\psi$. Then, $T \vdash \phi\,\mathtt{u}\,\psi$ or there exists a nonnegative integer m such that $T \vdash \bigwedge_{i=0}^{m}[\omega]^i\mathtt{g}(\phi \wedge \neg\psi) \wedge [\omega]^{m+1}(\phi\,\mathtt{u}\,\psi)$.*

Proof. Contrary to the assumption of the lemma, let $T \nvdash \phi\,\mathtt{u}\,\psi$ and

$$T \nvdash \bigwedge_{i=0}^{m}[\omega]^i\mathtt{g}(\phi \wedge \neg\psi) \wedge [\omega]^{m+1}\psi$$

for all $m \in \omega$. Since T is complete, $T \vdash \neg(\phi\,\mathtt{u}\,\psi)$ and $T \vdash \bigvee_{i=0}^{m}[\omega]^i\neg\mathtt{g}(\phi \wedge \neg\psi) \vee [\omega]^{m+1}\neg(\phi\,\mathtt{u}\,\psi)$ for all $m \in \omega$, so by R4 (for θ any axiom instance can be chosen), $T \vdash \neg(\phi\,\mathtt{U}\,\psi)$, which contradicts consistency of T. □

Lemma 3.2.5. *Let ϕ be any $L([1],[\omega])$-formula. Then, for all $\langle k,l\rangle \in \omega\times\omega$,*

$$\xi_T \models_{\langle k,l\rangle} \phi \text{ iff } T \vdash [\omega]^k[1]^l\phi.$$

Proof. The induction on the complexity of $L([1],[\omega])$-formulas will be used. The case of propositional letters is covered by definition of ξ_T.

$$\begin{aligned}
\xi_T \models_{\langle k,l\rangle} \neg\phi &\Leftrightarrow \xi_T \not\models_{\langle k,l\rangle} \phi\\
&\overset{IH}{\Leftrightarrow} T \nvdash [\omega]^k[1]^l\phi\\
&\Leftrightarrow T \vdash \neg[\omega]^k[1]^l\phi\\
&\overset{A3}{\Leftrightarrow} T \vdash [\omega]^k[1]^l\neg\phi;
\end{aligned}$$

$$\begin{aligned}
\xi_T \models_{\langle k,l\rangle} \phi\wedge\psi &\Leftrightarrow \xi_T \models_{\langle k,l\rangle} \phi \text{ and } \xi_T \models_{\langle k,l\rangle} \psi\\
&\overset{IH}{\Leftrightarrow} T \vdash [\omega]^k[1]^l\phi \text{ and } T \vdash [\omega]^k[1]^l\phi\\
&\Leftrightarrow T \vdash ([\omega]^k[1]^l\phi)\wedge([\omega]^k[1]^l\psi)\\
&\overset{A4}{\Leftrightarrow} T \vdash [\omega]^k[1]^l(\phi\wedge\psi);
\end{aligned}$$

$$\begin{aligned}
\xi_T \models_{\langle k,l\rangle} [1]\phi &\Leftrightarrow \xi_T \models_{\langle k,l+1\rangle} \phi\\
&\overset{IH}{\Leftrightarrow} T \vdash [\omega]^k[1]^{l+1}\phi\\
&\Leftrightarrow T \vdash [\omega]^k[1]^l[1]\phi;
\end{aligned}$$

$$\begin{aligned}
\xi_T \models_{\langle k,l\rangle} [\omega]\phi &\Leftrightarrow \xi_T \models_{\langle k+1,0\rangle} \phi\\
&\overset{IH}{\Leftrightarrow} T \vdash [\omega]^{k+1}\phi\\
&\Leftrightarrow T \vdash [\omega]^k[\omega]\phi\\
&\overset{A2}{\Leftrightarrow} T \vdash [\omega]^k[1]^l[\omega]\phi.
\end{aligned}$$

□

Lemma 3.2.6. *Let ϕ be any $L([1],[\omega],\mathfrak{u})$-formula. Then, for all $\langle k,l\rangle \in \omega\times\omega$,*

$$\xi_T \models_{\langle k,l\rangle} \phi \text{ iff } T \vdash [\omega]^k[1]^l\phi.$$

Proof. As in the proof of the previous lemma, the induction on the complexity of $L([1],[\omega],\mathfrak{u})$-formulas will be used. By Lemma 3.2.5, the only thing that has to be considered is the case of $\mathfrak{u}$-formulas.

Let $\xi_T \models_{\langle k,l\rangle} \phi\,\mathfrak{u}\,\psi$. Then, $\xi_T \models_{\langle k,l\rangle} \psi$ or there exists a nonnegative integer m such that $\xi_T \models_{\langle k,l\rangle} \bigwedge_{i=0}^{m}[1]^i(\phi\wedge\neg\psi)\wedge[1]^{m+1}\psi$. By induction hypothesis, $T \vdash [\omega]^k[1]^l\psi$ or $T \vdash [\omega]^k[1]^l\bigwedge_{i=0}^{m}[1]^i(\phi\wedge\neg\psi)\wedge[1]^{m+1}\psi$. By A4, A5 and A7, $T \vdash [\omega]^k[1]^l(\phi\,\mathfrak{u}\,\psi)$.

Conversely, let $T \vdash [\omega]^k[1]^l(\phi\,\mathfrak{u}\,\psi)$. By A4 and Lemma 3.2.3, $T \vdash [\omega]^k[1]^l\psi$ or there exists a nonnegative integer m such that $T \vdash [\omega]^k[1]^l\bigwedge_{i=0}^{m}[1]^i(\phi\wedge\neg\psi)\wedge[1]^{m+1}\psi$. By induction hypothesis, $\xi_T \models_{\langle k,l\rangle} \psi$ or $\xi_T \models_{\langle k,l\rangle} \bigwedge_{i=0}^{m}[1]^i(\phi\wedge\neg\psi)\wedge[1]^{m+1}\psi$, so by definition of $\models$, $\xi_T \models_{\langle k,l\rangle} \phi\,\mathfrak{u}\,\psi$. □

Lemma 3.2.7. *Let ϕ be any $L([1],[\omega],\mathtt{u},\mathtt{U})$-formula. Then, for all $\langle k,l\rangle \in \omega\times\omega$,*

$$\xi_T \models_{\langle k,l\rangle} \phi \text{ iff } T \vdash [\omega]^k[1]^l\phi.$$

Proof. A straightforward modification of the proof of the previous lemma based on lemmas 3.2.4 and 3.2.6. □

Corollary 3.2.1. *$\xi_T \models_{\langle 0,0\rangle} \phi$ iff $T \vdash \phi$.*

Proof. An immediate consequence of Lemma 3.2.7. □

Hence, the fact that each consistent theory is satisfiable has been proved, which concludes the proof of strong completeness theorem for $L([1],[\omega],\mathtt{u},\mathtt{U})$.

Representation of zero-time transitions

The main technical idea behind choosing ω^2 as a model of the time flow is to separate states of the system and transitions from one state to the next. More precisely, time instants of the form $\langle k,0\rangle$ are reserved for temporal description of different states of a system, while transitions from $\langle k,0\rangle$ to $\langle k+1,0\rangle$ are temporally described throughout the k-th ω-stick $\{\langle k,n\rangle \mid n\in\omega\}$. Recall that the state of a system in any time instant $\mathbf{t}$ is described by formulas satisfied in $\mathbf{t}$.

As it was described in [44], any transition can occur within a closed time interval. In order to model this phenomenon, the following two things have to be provided:

- Any formula ϕ that is satisfied in all $\langle k,n\rangle$, $n \geqslant n_0$, should also be satisfied in $\langle k+1,0\rangle$;
- Any change can occur only once.

This leads to additional transition axioms:

TR1 $\mathtt{g}\phi \to [\omega]\phi$;

TR2 $(\phi \wedge [1]\neg\phi) \to \mathtt{g}[1]\neg\phi$.

Clearly, the strong completeness theorem holds for the extended system as well since both TR1 and TR2 can be seen as $L([1],[\omega],\mathtt{u},\mathtt{U})$-theories.

3.2.3 Decidability

In the process of showing decidability of the SAT problem for $L([1],[\omega],\mathtt{u},\mathtt{U})$-formulas and determination of the complexity, the argument presented by A. P. Sistla and E. M. Clarke in [75] will be modified.

By $Var(\phi)$ will be denoted the set of all propositional variables appearing in ϕ, while by $\mathrm{Sub}(\phi)$ will be denoted the set of all subformulas of ϕ. Note that

$$|\mathrm{Sub}(\phi)| \leqslant \mathrm{length}(\phi),$$

where $|\mathrm{Sub}(\phi)|$ is the cardinal number of $\mathrm{Sub}(\phi)$ and $\mathrm{length}(\phi)$ is the number of symbols in ϕ. Moreover, let

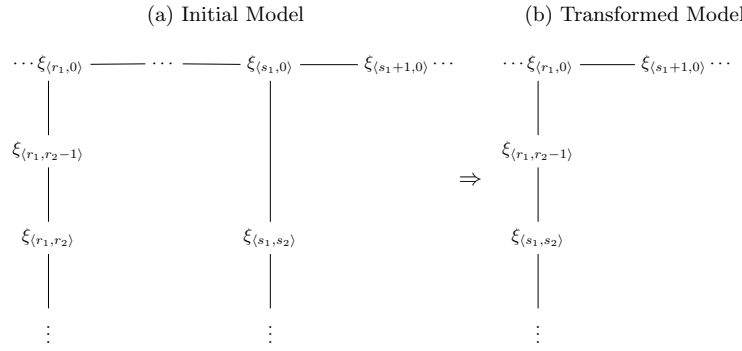


Figure 3.5: $r \sim s$

$$\mathrm{Sub}^*(\phi) = \mathrm{Sub}(\phi) \cup \{\psi \,\mathtt{u}\, \theta \;:\; \psi \,\mathtt{U}\, \theta \in \mathrm{Sub}(\phi)\}$$

and

$$\mathrm{Sub}^*(\phi, \mathbf{t}, \xi) = \{\psi \in \mathrm{Sub}^*(\phi) \mid \xi \models_{\mathbf{t}} \psi\}.$$

Note that $|\mathrm{Sub}^*(\phi)| \leqslant 2\,\mathrm{length}(\phi)$.

For the given formula ϕ and any model $\xi : \omega \times \omega \times Var \longrightarrow \{0,1\}$ a binary relation $\sim_{\phi,\xi}$ on $\omega \times \omega$ is defined as follows:

$$\mathbf{r} \sim_{\phi,\xi} \mathbf{s} \quad \text{iff} \quad \mathrm{Sub}^*(\phi, \mathbf{r}, \xi) = \mathrm{Sub}^*(\phi, \mathbf{s}, \xi).$$

If the context is clear, ϕ and ξ will be omitted from the subscript and just $\sim$ will be written instead of $\sim_{\phi,\xi}$. It is easy to see that $\sim$ is an equivalence relation with no more than $4^{\mathrm{length}(\phi)}$ classes.

In the sequel, following the procedure from [75], it will be shown that a model satisfying a formula can be transformed to another model of the formula which is more suitable to prove decidability. In this transformation the initial model will be pruned of non-essential worlds. Figure 3.5 illustrates the deletion process.

Lemma 3.2.8, also known as deleting or filtration lemma, is the main technical tool for the removal of the redundant worlds from the given model. More precisely, Lemma 3.2.8 says that if $\mathbf{r} < \mathbf{s}$ and $\mathbf{r} \sim_{\phi,\xi} \mathbf{s}$, then ξ-values on $[\mathbf{r},\mathbf{s})_{\mathrm{lex}} \times Var$ do not have any impact on $\mathrm{Sub}^*(\phi, \mathbf{t}, \xi)$ for all $\mathbf{t} \notin [\mathbf{r},\mathbf{s})_{\mathrm{lex}}$. In particular, if $\mathbf{t} < \mathbf{r} < \mathbf{s}$ and $\mathbf{r} \sim_{\phi,\xi} \mathbf{s}$, then the truth value of the predicate $\xi \models_{\mathbf{t}} \phi$ is independent of $\xi|_{[\mathbf{r},\mathbf{s})_{\mathrm{lex}} \times Var}$. Moreover, there is a unique isomorphism

$$f : \langle \omega \times \omega, \leqslant_{\mathrm{lex}} \rangle \cong \langle (\omega \times \omega) \setminus [\mathbf{r},\mathbf{s})_{\mathrm{lex}}, \leqslant_{\mathrm{lex}} \rangle,$$

so if the model η is defined by $\eta(i,j,p) = \xi(f(i,j),p)$ for all $p \in Var$, then the following holds:

$$\xi \models_{\mathbf{t}} \phi \quad \text{iff} \quad \eta \models_{\mathbf{t}} \phi.$$

Note that η is just a technical rearrangement of $\xi \setminus \xi|_{[\mathbf{r},\mathbf{s})_{\mathrm{lex}} \times Var}$ and that the true meaning of the previous equivalence is

$$\xi \models_{\mathbf{t}} \phi \quad \text{iff} \quad \xi \setminus \xi|_{[\mathbf{r},\mathbf{s})_{\mathrm{lex}} \times Var} \models_{\mathbf{t}} \phi,$$

which is the reason why Lemma 3.2.8 is referred as deleting or filtration lemma.

Lemma 3.2.8. *With notation as before, suppose that ϕ, ξ, η, $\mathbf{r}$ and $\mathbf{s}$ satisfy the following conditions:*

1. $\mathbf{r} < \mathbf{s}$;
2. $\mathrm{Sub}^*(\phi, \mathbf{r}, \xi) = \mathrm{Sub}^*(\phi, \mathbf{s}, \xi)$;
3. $\eta(\mathbf{t}, p) = \xi(\mathbf{t}, p)$ *for all* $\mathbf{t} < \mathbf{r}$ *and all* $p \in Var$;
4. $\eta(r_1 + i, r_2 + j, p) = \xi(s_1 + i, s_2 + j, p)$ *for all* $i, j \in \omega$ *and all* $p \in Var$.

Then, the following hold:

(a) $\mathrm{Sub}^*(\phi, \mathbf{t}, \xi) = \mathrm{Sub}^*(\phi, \mathbf{t}, \eta)$ *for all* $\mathbf{t} < \mathbf{r}$;

(b) $\mathrm{Sub}^*(\phi, \langle s_1 + i, s_2 + j\rangle, \xi) = \mathrm{Sub}^*(\phi, \langle r_1 + i, r_2 + j\rangle, \eta)$ *for all* $i, j \in \omega$.

Proof. Note that (b) is an immediate consequence of Definition 3.2.2 and the fourth condition in the statement of the lemma. Therefore, it remains to prove (a). It turns out that a somewhat stronger claim is easier to prove. Namely, let A be the set of all formulas with variables from $Var(\phi)$ that satisfy (a) then $Var(\phi) \subseteq A$ and A is closed under $\neg$, $\wedge$, $[1]$, $[\omega]$, $\mathtt{u}$ and $\mathtt{U}$. Moreover, since

$$\psi \,\mathtt{u}\, \theta \Leftrightarrow \theta \vee \bigvee_{n\in\omega} \bigwedge_{k=0}^{n} [1]^k(\psi \wedge \neg\theta) \wedge [1]^{n+1}\theta$$

and

$$\psi \,\mathtt{U}\, \theta \Leftrightarrow \psi \,\mathtt{u}\, \theta \vee \bigvee_{n\in\omega} \bigwedge_{k=0}^{n} [\omega]^k \mathtt{g}(\psi \wedge \neg\theta) \wedge [\omega]^{n+1}(\psi \,\mathtt{u}\, \theta),$$

for the verification of the lemma it is sufficient to prove that $Var(\phi) \subseteq A$ and that A is closed under $\neg$, $\wedge$, $[1]$ and $[\omega]$.

By (3), all propositional letters form $Var(\phi)$ satisfy (a). Moreover, if $\neg\psi, \theta_1 \wedge \theta_2 \in \mathrm{Sub}(\phi)$ and ψ, θ_1 and θ_2 satisfy (a), then by Definition 3.2.2 it immediately follows that $\neg\psi$ and $\theta_1 \wedge \theta_2$ also satisfy (a).

Suppose that $[1]\psi \in \mathrm{Sub}^*(\phi)$ and that ψ satisfies (a), then $\xi \models_{\mathbf{t}} [1]\psi$ iff $\eta \models_{\mathbf{t}} [1]\psi$ for all $\mathbf{t} < \mathbf{r}$ has to be proved. There are two relevant cases:

- $\langle t_1, t_2 + 1\rangle < \mathbf{r}$. Then,

$$\begin{aligned} \xi \models_{\mathbf{t}} [1]\psi &\Leftrightarrow \xi \models_{\langle t_1, t_2+1\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle t_1, t_2+1\rangle} \psi \quad (\psi \in A \text{ and } \langle t_1, t_2 + 1\rangle < \mathbf{r}) \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [1]\psi. \end{aligned}$$

- $\langle t_1, t_2 + 1\rangle = \mathbf{r}$ (note that this case cannot occur if $\mathbf{r}$ is a limit, i.e. if $r_2 = 0$). Then,

$$\begin{aligned} \xi \models_{\mathbf{t}} [1]\psi &\Leftrightarrow \xi \models_{\mathbf{r}} \psi \\ &\Leftrightarrow \xi \models_{\mathbf{s}} \psi \quad (\text{follows from}(2)) \\ &\Leftrightarrow \eta \models_{\mathbf{r}} \psi \quad (\text{follows from}(4)) \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [1]\psi. \end{aligned}$$

Suppose that $[\omega]\,\psi \in \mathrm{Sub}^*(\phi)$ and that ψ satisfies (a). There are three relevant cases:

- $\langle t+1,0\rangle < \mathbf{r}$. Then,

$$\begin{aligned}\xi \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \xi \models_{\langle t_1+1,0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle t_1+1,0\rangle} \psi \quad (\psi \in A \text{ and } \langle t_1+1,0\rangle < \mathbf{r}) \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [\omega]\psi;\end{aligned}$$

- Let $\langle t_1+1,0\rangle = \mathbf{r}$. Then,

$$\begin{aligned}\xi \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \xi \models_{\mathbf{r}} \psi \\ &\Leftrightarrow \xi \models_{\mathbf{s}} \psi \quad \text{(follows from (2))} \\ &\Leftrightarrow \eta \models_{\mathbf{r}} \psi \quad \text{(follows from (4))} \\ &\Leftrightarrow \eta \models_{\langle t_1+1,0\rangle} \psi \quad (\langle t_1+1,0\rangle = \mathbf{r}) \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [\omega]\psi;\end{aligned}$$

- Let $\langle t_1+1,0\rangle > \mathbf{r}$. Since $\mathbf{t} < \mathbf{r}$, it must be $t_1 = r_1$, so

$$\begin{aligned}\xi \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \xi \models_{\mathbf{r}} [\omega]\psi \\ &\Leftrightarrow \xi \models_{\mathbf{s}} [\omega]\psi \quad \text{(follows from (2))} \\ &\Leftrightarrow \xi \models_{\langle s_1+1,0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle r_1+1,0\rangle} \psi \quad \text{(follows from (4))} \\ &\Leftrightarrow \eta \models_{\langle t_1+1,0\rangle} \psi \quad (r_1 = t_1) \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [\omega]\psi.\end{aligned}$$

□

Definition 3.2.6. *Suppose that* $\mathbf{r} < \mathbf{s}$ *and* $\xi \models_{\mathbf{r}} \psi \,\mathtt{U}\, \theta$. $\psi \,\mathtt{U}\, \theta$ *is fulfilled before* $\mathbf{s}$ *iff there exists* $\mathbf{t} \in [\mathbf{r},\mathbf{s})_{\mathrm{lex}}$ *such that* $\xi \models_{\mathbf{t}} \theta$ *and* $\xi \models_{\mathbf{t}'} \psi$ *for all* $\mathbf{t}'$ *such that* $\mathbf{r} \leqslant \mathbf{t}' < \mathbf{t}$.

In Lemma 3.2.9 it is shown that if $\langle k,0\rangle \sim_{\phi,\xi} \langle k+m,0\rangle$ and all U-formulas from $\mathrm{Sub}^*(\phi,\langle k,0\rangle,\xi)$ are fulfilled before $\langle k+m,0\rangle$, then $\mathrm{Sub}^*(\phi,\mathbf{r},\xi)$ does not depend on the ξ-values on $\{\langle \mathbf{t},p\rangle \;:\; \mathbf{t} \geqslant \langle k+m,0\rangle,\ p \in Var\}$ for all $\mathbf{r} < \langle k+m,0\rangle$. In particular, for any $\mathbf{r} \leqslant \langle k,0\rangle$ the following holds:

$$\xi \models_{\mathbf{r}} \phi \quad \text{iff} \quad \xi|_{[\mathbf{r},\langle k+m,0\rangle)_{\mathrm{lex}}\times Var} \models_{\mathbf{r}} \phi.$$

The technical inconvenience lies in the fact that the function $\xi|_{[\mathbf{r},\langle k+m,0\rangle)_{\mathrm{lex}}\times Var}$ is not a model, so its domain has to be extended to $\omega\times\omega\times Var$. The natural way to do so is to extend periodically $\xi|_{[\mathbf{r},\langle k+m,0\rangle)_{\mathrm{lex}}\times Var}$ to the ultimately periodic model η with the outer period (outer loop) m. More precisely, η coincides with ξ on the set $[\langle 0,0\rangle,\langle k+m,0\rangle)_{\mathrm{lex}} \times Var$ and $\eta(k+i,j,p) = \eta(k+m+i,j,p)$ for all $i,j \in \omega$ and all $p \in Var$.

Lemma 3.2.9 (Outer loop)**.** *With notation as before, suppose that* ϕ, ξ, η, k *and* m *(*$k,m \in \omega$ *and* $m > 0$*) satisfy the following conditions:*

1. $\mathrm{Sub}^*(\phi,\langle k,0\rangle,\xi) = \mathrm{Sub}^*(\phi,\langle k+m,0\rangle,\xi)$;

2. $\xi(\mathbf{t}, p) = \eta(\mathbf{t}, p)$ *for all* $\mathbf{t} < \langle k+m, 0\rangle$ *and all* $p \in Var$;

3. $\eta(\mathbf{s}, p) = \eta(\langle s_1 + m, s_2\rangle, p)$ *for all* $\mathbf{s} \geqslant \langle k, 0\rangle$ *and all* $p \in Var$;

4. *Each* $\psi \,\mathtt{U}\, \theta \in \mathrm{Sub}^*(\phi, \langle k, 0\rangle, \xi)$ *is fulfilled before* $\langle k+m, 0\rangle$.

Then, the following hold:

(a) $\mathrm{Sub}^*(\phi, \mathbf{t}, \xi) = \mathrm{Sub}^*(\phi, \mathbf{t}, \eta)$ *for all* $\mathbf{t} < \langle k+m, 0\rangle$;

(b) $\mathrm{Sub}^*(\phi, \mathbf{s}, \eta) = \mathrm{Sub}^*(\phi, \langle s_1 + m, s_2\rangle, \eta)$ *for all* $\mathbf{s} \geqslant \langle k, 0\rangle$.

Proof. Both cases (a) and (b) will be proved simultaneously by induction on the complexity of formulas. Due to the third condition of the lemma, which establishes periodicity of η, to prove the statement (b) it is sufficient to consider only $\langle k, 0\rangle \leqslant \mathbf{s} < \langle k+m, 0\rangle$. Firstly, (a) and (b) are obviously true for all propositional variables from $Var(\phi)$ and they are preserved by negation and conjunction (this is an immediate consequence of Definition 3.2.2).

Suppose that $[1]\psi \in \mathrm{Sub}^*(\phi)$ and that ψ satisfies both (a) and (b). Let $\mathbf{t} < \langle k+m, 0\rangle$. Then,

$$\begin{aligned}
\xi \models_{\mathbf{t}} [1]\psi &\Leftrightarrow \xi \models_{\langle t_1, t_2+1\rangle} \psi \\
&\Leftrightarrow \eta \models_{\langle t_1, t_2+1\rangle} \psi \quad (\langle t_1, t_2+1\rangle < \langle k+m, 0\rangle \text{ and } \psi \text{ satisfies } (a)) \\
&\Leftrightarrow \eta \models_{\mathbf{t}} [1]\psi.
\end{aligned}$$

Let $\langle k, 0\rangle \leqslant \mathtt{t} < \langle k+m, 0\rangle$. Then,

$$\begin{aligned}
\eta \models_{\mathbf{t}} [1]\psi &\Leftrightarrow \eta \models_{\langle t_1, t_2+1\rangle} \psi \\
&\Leftrightarrow \eta \models_{\langle t_1+m, t_2+1\rangle} \psi \quad (\langle t_1, t_2+1\rangle < \langle k+m, 0\rangle \text{ and } \psi \text{ satisfies } (b)) \\
&\Leftrightarrow \eta \models_{\langle t_1+m, t_2\rangle} [1]\psi.
\end{aligned}$$

Suppose that $[\omega]\psi \in \mathrm{Sub}^*(\phi)$ and that ψ satisfies both (a) and (b). Let $\mathbf{t} < \langle k+m, 0\rangle$. If $\langle t_1+1, 0\rangle < \langle k+m, 0\rangle$, then

$$\begin{aligned}
\xi \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \xi \models_{\langle t_1+1, 0\rangle} \psi \\
&\Leftrightarrow \eta \models_{\langle t_1+1, 0\rangle} \psi \\
&\Leftrightarrow \eta \models_{\mathbf{t}} [\omega]\psi.
\end{aligned}$$

If $\langle t_1+1, 0\rangle = \langle k+m, 0\rangle$, then

$$\begin{aligned}
\xi \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \xi \models_{\langle t_1+1, 0\rangle} \psi \\
&\Leftrightarrow \xi \models_{\langle k+m, 0\rangle} \psi \\
&\Leftrightarrow \xi \models_{\langle k, 0\rangle} \psi \quad \text{(by (1))} \\
&\Leftrightarrow \eta \models_{\langle k, 0\rangle} \psi \quad \text{(by (2))} \\
&\Leftrightarrow \eta \models_{\langle k+m, 0\rangle} \psi \quad \text{(by (b))} \\
&\Leftrightarrow \eta \models_{\mathbf{t}} [\omega]\psi.
\end{aligned}$$

Let $\langle k, 0\rangle \leqslant \mathbf{t} < \langle k+m, 0\rangle$. If $\langle t_1+1, 0\rangle < \langle k+m, 0\rangle$, then

$$\begin{aligned}
\eta \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \eta \models_{\langle t_1+1, 0\rangle} \psi \\
&\Leftrightarrow \eta \models_{\langle t_1+1+m, 0\rangle} \psi \\
&\Leftrightarrow \eta \models_{\langle t_1+m+1, 0\rangle} \psi \\
&\Leftrightarrow \eta \models_{\langle t_1+m, t_2\rangle} [\omega]\psi.
\end{aligned}$$

If $\langle t_1+1,0\rangle = \langle k+m,0\rangle$, then

$$\begin{aligned}\eta \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \eta \models_{\langle t_1+1,0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle k+m,0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle k+2m,0\rangle} \psi \quad \text{(by (3))} \\ &\Leftrightarrow \eta \models_{\langle t_1+m+1,0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle t_1+m,t_2\rangle} [\omega]\psi.\end{aligned}$$

Suppose that $\psi \,\mathtt{u}\, \theta \in \mathrm{Sub}^*(\phi)$ and that ψ and θ satisfy both (a) and (b). Let $\mathbf{t} < \langle k+m,0\rangle$. Then, $\langle t_1, t_2+i\rangle < \langle k+m,0\rangle$ for all $i < \omega$, so $\xi \models_{\langle t_1,t_2+i\rangle} \theta$ iff $\eta \models_{\langle t_1,t_2+i\rangle} \theta$ for all $i < \omega$ and $\xi \models_{\langle t_1,t_2+i\rangle} \psi \wedge \neg\theta$ iff $\eta \models_{\langle t_1,t_2+i\rangle} \psi \wedge \neg\theta$ for all $i < \omega$. Consequently, $\xi \models_{\mathbf{t}} \psi \,\mathtt{u}\, \theta$ iff $\eta \models_{\mathbf{t}} \psi \,\mathtt{u}\, \theta$. Similarly, if $\langle k,0\rangle \leqslant \mathbf{t} < \langle k+m,0\rangle$, then $\eta \models_{\mathbf{t}} \psi \,\mathtt{u}\, \theta$ iff $\eta \models_{\langle t_1+m,t_2\rangle} \psi \,\mathtt{u}\, \theta$

Suppose that $\psi \,\mathtt{U}\, \theta \in \mathrm{Sub}^*(\phi)$ and that ψ and θ satisfy both (a) and (b). Then, for all $\mathbf{t} < \langle k+m,0\rangle$, $\xi \models_{\mathbf{t}} \theta$ iff $\eta \models_{\mathbf{t}} \theta$ and $\xi \models_{\mathbf{t}} \psi \wedge \neg\theta$ iff $\xi \models_{\mathbf{t}} \psi \wedge \neg\theta$.

Let $\xi \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$ for some $\mathbf{t} < \langle k+m,0\rangle$. By (4), there exists $\mathbf{t} \leqslant \mathbf{r} < \langle k+m,0\rangle$ such that $\xi \models_{\mathbf{r}} \theta$. Since $[\mathbf{t}, \langle k+m,0\rangle)_{\mathrm{lex}}$ is a well ordering, it is possible to chose the minimal $\mathbf{r} \in [\mathbf{t}, \langle k+m,0\rangle)_{\mathrm{lex}}$ such that $\xi \models_{\mathbf{r}} \theta$. Now $\xi \models_{\mathbf{s}} \psi \wedge \neg\theta$ holds for all $\mathbf{t} \leqslant \mathbf{s} < \mathbf{r}$, so by induction hypothesis and Definition 3.2.2, $\eta \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$.

Let $\eta \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$ for some $\mathbf{t} < \langle k+m,0\rangle$. Then, there exists $\mathbf{r} \geqslant \mathbf{t}$ such that $\eta \models_{\mathbf{r}} \theta$ and $\eta \models_{\mathbf{s}} \psi \wedge \neg\theta$ for all $\mathbf{t} \leqslant \mathbf{s} < \mathbf{r}$. Since $[\mathbf{t}, \infty)_{\mathrm{lex}}$ is a well ordering, it is possible to chose the minimal $\mathbf{r}$. Moreover, (b) implies that the minimal $\mathbf{r}$ is strictly less than $\langle k+m,0\rangle$, so $\xi \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$ holds.

It remains to prove (b) for $\psi \,\mathtt{U}\, \theta$, provided that ψ and θ satisfy both (a) and (b). By (b) and Definition 3.2.2, $\eta \models_{\mathbf{s}} \theta$ iff $\eta \models_{\langle s_1+m,s_2\rangle} \theta$ and $\eta \models_{\mathbf{s}} \psi \wedge \neg\theta$ iff $\eta \models_{\langle s_1+m,s_2\rangle} \psi \wedge \neg\theta$ for all $\mathbf{s} \in [\langle k,0\rangle, \langle k+m,0\rangle)_{\mathrm{lex}}$. As it has already been proved, $\eta \models_{\mathbf{s}} \psi \,\mathtt{U}\, \theta$ iff $\xi \models_{\mathbf{s}} \psi \,\mathtt{U}\, \theta$ for all $\mathbf{s} \in [\langle k,0\rangle, \langle k+m,0\rangle)_{\mathrm{lex}}$. As a consequence, η satisfies (4).

Let $\langle k,0\rangle \leqslant \mathbf{t} < \langle k+m,0\rangle$ and let $\eta \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$. By (4), there exists $\mathbf{t} \leqslant \mathbf{r} < \langle k+m,0\rangle$ such that $\eta \models_{\mathbf{r}} \theta$ and $\eta \models_{\mathbf{s}} \psi \wedge \neg\theta$ for all $\mathbf{s} \in [\mathbf{t},\mathbf{r})_{\mathrm{lex}}$. By induction hypothesis, $\eta \models_{\langle r_1+m,r_2\rangle} \theta$ and $\eta \models_{\langle s_1+m,s_2\rangle} \psi \wedge \neg\theta$ for all $\mathbf{s} \in [\mathbf{t},\mathbf{r})_{\mathrm{lex}}$, so $\eta \models_{\langle t_1+m,t_2\rangle} \psi \,\mathtt{U}\, \theta$. The converse implication can be proved similarly. □

Definition 3.2.7. *Let $j < k$ and $\xi \models_{\langle i,j\rangle} \psi \,\mathtt{u}\, \theta$. $\psi \,\mathtt{u}\, \theta$ is fulfilled before $\langle i,k\rangle$ iff there exists $r \in \omega$ such that $j \leqslant r < k$, $\xi_{\langle i,r\rangle} \models \theta$ and $\xi \models_{\langle i,s\rangle} \psi$ for all $s \in \omega$ such that $j \leqslant s < r$.*

Lemma 3.2.10 is the local variant of Lemma 3.2.9. More precisely, if $\langle k,l\rangle \sim_{\phi,\xi} \langle k,l+m\rangle$ and all $\mathtt{u}$-formulas from $\mathrm{Sub}^*(\phi, \langle k,l\rangle, \xi)$ are fulfilled before $\langle k,l+m\rangle$, then $\mathrm{Sub}^*(\phi,\mathbf{r},\xi)$ does not depend on the ξ-values on $[\langle k,l+m\rangle, \langle k+1,0\rangle)_{\mathrm{lex}} \times Var$ for all $\mathbf{r} \notin [\langle k,l+m\rangle, \langle k+1,0\rangle)_{\mathrm{lex}} \times Var$. In particular, for any $\mathbf{r} < \langle k,l+m\rangle$ the following holds

$$\xi \models_{\mathbf{r}} \phi \quad \text{iff} \quad \xi|_{\left((\omega\times\omega)\setminus[\langle k,l+m\rangle,\langle k+1,0\rangle)_{\mathrm{lex}}\right)\times Var} \models_{\mathbf{r}} \phi.$$

Similarly as before, technical difficulty lies in the fact that the function

$$\xi|_{\left((\omega\times\omega)\setminus[\langle k,l+m\rangle,\langle k+1,0\rangle)_{\mathrm{lex}}\right)\times Var}$$

is not a model. The natural way to extend its domain to $\omega \times \omega \times Var$ is to introduce m as the local period (inner loop, period on the second coordinates) on the $\{k\} \times \omega \times Var$. In particular, the extension η coincides with ξ on the set

$$\big((\omega \times \omega) \setminus [\langle k, l+m\rangle, \langle k+1, 0\rangle)_{\text{lex}}\big) \times Var,$$

and $\eta(k, l+i, p) = \eta(k, l+m+i, p)$ for all $i \in \omega$ and all $p \in Var$.

Lemma 3.2.10 (Inner loop)**.** *With notation as before, suppose that ϕ, ξ, η, k, l and m ($k, l, m \in \omega$ and $m > 0$) satisfy the following conditions:*

1. $\mathrm{Sub}^*(\phi, \langle k, l\rangle, \xi) = \mathrm{Sub}^*(\phi, \langle k, l+m\rangle, \xi)$;
2. $\xi(\mathbf{t}, p) = \eta(\mathbf{t}, p)$ *for all* $\mathbf{t} < \langle k, l+m\rangle$, *all* $\mathbf{t} \geqslant \langle k+1, 0\rangle$ *and all* $p \in Var$;
3. $\eta(k, i, p) = \eta(k, i+m, p)$ *for all* $i \geqslant l$ *and all* $p \in Var$;
4. *All* $\psi \,\mathbf{u}\, \theta \in \mathrm{Sub}^*(\phi, \langle k, l\rangle, \xi)$ *are fulfilled before* $\langle k, l+m\rangle$.

Then, the following hold:

(a) $\mathrm{Sub}^*(\phi, \mathbf{t}, \xi) = \mathrm{Sub}^*(\phi, \mathbf{t}, \eta)$ *for all* $\mathbf{t} < \langle k, l+m\rangle$ *and all* $\mathbf{t} \geqslant \langle k+1, 0\rangle$;

(b) $\mathrm{Sub}^*(\phi, \mathbf{s}, \eta) = \mathrm{Sub}^*(\phi, \langle s_1, s_2+m\rangle, \eta)$ *for all* $\mathbf{s} \in [\langle k, l\rangle, \langle k+1, 0\rangle)_{\text{lex}}$.

Proof. Both cases (a) and (b) will be proved simultaneously by induction on the complexity of formulas. As before, the statement is obviously true for propositional variables and its validity is preserved under negation and conjunction. Furthermore, (2) implies that $\mathrm{Sub}^*(\phi, \mathbf{t}, \xi) = \mathrm{Sub}^*(\phi, \mathbf{t}, \eta)$ for all $\mathbf{t} \geqslant \langle k+1, 0\rangle$. Similarly as in the proof of Lemma 3.2.9 it can be shown that (4) holds for η as well.

Suppose that $[1]\psi \in \mathrm{Sub}^*(\phi)$ and that ψ satisfies both (a) and (b). Let $\mathbf{t} < \langle k, l+m\rangle$. If $t_2 + 1 < l + m$, then by induction hypothesis $\xi \models_{\mathbf{t}} [1]\psi$ iff $\eta \models_{\mathbf{t}} [1]\psi$. Let $t_2 + 1 = l + m$. Then

$$\begin{aligned} \xi \models_{\mathbf{t}} [1]\psi &\Leftrightarrow \xi \models_{\langle k, l+m\rangle} \psi \\ &\Leftrightarrow \xi \models_{\langle k, l\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle k, l+m\rangle} \psi \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [1]\psi. \end{aligned}$$

Let $\langle k, l\rangle \leqslant \mathbf{t} < \langle k+1, 0\rangle$. Then,

$$\begin{aligned} \eta \models_{\mathbf{t}} [1]\psi &\Leftrightarrow \eta \models_{\langle t_1, t_2+1\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle t_1, t_2+1+m\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle t_1, t_2+m+1\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle t_1, t_2+m\rangle} [1]\psi. \end{aligned}$$

Suppose that $[\omega]\psi \in \mathrm{Sub}^*(\phi)$ and that ψ satisfies (a) and (b). Let $\mathbf{t} < \langle k, l+m\rangle$. If $t_1 + 1 < k$, then (a) can be straightforwardly verified for $[\omega]\psi$ by induction hypothesis. If $t_1 + 1 = k$, then

$$\begin{aligned} \xi \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \xi \models_{\langle k, 0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle k, 0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [\omega]\psi. \end{aligned}$$

Let $t_1 = k$. Then,

$$\begin{aligned} \xi \models_{\mathbf{t}} [\omega]\psi &\Leftrightarrow \xi \models_{\langle k+1,0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\langle k+1,0\rangle} \psi \\ &\Leftrightarrow \eta \models_{\mathbf{t}} [\omega]\psi. \end{aligned}$$

Let $\langle k,l\rangle \leqslant \mathbf{t} < \langle k+1,0\rangle$. Then, by Definition 3.2.2, $\eta \models_{\mathbf{t}} [\omega]\psi$ iff $\eta \models_{\langle k,l+i\rangle} [\omega]\psi$ for all $i < \omega$. Hence $[\omega]\psi$ satisfies (b).

Suppose that $\psi \,\mathtt{u}\, \theta \in \mathrm{Sub}^*(\phi)$ and that both ψ and θ satisfy (a) and (b). Let $\mathbf{t} < \langle k, l+m\rangle$ and $\xi \models_{\mathbf{t}} \psi \,\mathtt{u}\, \theta$. The only nontrivial case is when $t_1 = k$ and $t_2 \geqslant l$. Then, there exists $n < \omega$ such that $t_2 + n < l + m$, $\xi_{\langle k,t_2+n\rangle} \models \theta$ and $\xi_{\langle k,l+i\rangle}\psi \wedge \neg\theta$ for all $i \in \{0,\ldots,n-1\}$. By induction hypothesis, ξ can be replaced with η, so $\eta \models_{\mathbf{t}} \psi \,\mathtt{u}\, \theta$. The converse implication can be proved similarly.

Let $\langle k,l\rangle \leqslant \mathbf{t} < \langle k+1,0\rangle$. By induction hypothesis, for all $i < \omega$, $\eta \models_{\langle k,i\rangle} \theta$ holds iff $\eta \models_{\langle k,i+m\rangle} \theta$ and $\eta \models_{\langle k,i\rangle} \psi \wedge \neg\theta$ holds iff $\eta \models_{\langle k,i+m\rangle} \psi \wedge \neg\theta$. Hence, by Definition 3.2.2, $\psi \,\mathtt{u}\, \theta$ satisfies (b).

Suppose that $\psi \,\mathtt{U}\, \theta \in \mathrm{Sub}^*(\phi)$ and that both ψ and θ satisfy (a) and (b). Let $\mathbf{t} < \langle k, l+m\rangle$ and $\xi \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$. Then, there exists $\mathbf{r} \geqslant \mathbf{t}$ such that $\xi \models_{\mathbf{r}} \theta$ and $\xi \models_{\mathbf{s}} \psi \wedge \neg\theta$ for all $\mathbf{s}$ such that $\mathbf{t} \leqslant \mathbf{s} < \mathbf{r}$. There are four relevant possibilities:

- $\mathbf{r} < \langle k, l+m\rangle$,
- $\langle k, l+m\rangle \leqslant \mathbf{r} < \langle k+1,0\rangle$, and $\mathbf{t} < \langle k,l\rangle$
- $\langle k, l+m\rangle \leqslant \mathbf{r} < \langle k+1,0\rangle$, and $\langle k,l\rangle \leqslant \mathbf{t} < \langle k,l+m\rangle$
- $\langle k+1,0\rangle \leqslant \mathbf{r}$.

In the first case, since ξ and η coincide for $\mathbf{t} < \langle k, l+m\rangle$, $\xi \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta \Leftrightarrow \eta \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$ holds.

In the second case, it must be $\xi \models_{\langle k,l\rangle} \psi \,\mathtt{U}\, \theta$, and since $\mathbf{r} < \langle k+1,0\rangle$, $\xi \models_{\langle k,l\rangle} \psi \,\mathtt{u}\, \theta$ holds. Due to (4), $\mathbf{r} < \langle k, l+m\rangle$ holds, which contradicts the assumed $\mathbf{r} \geqslant \langle k, l+m\rangle$.

In the third case, $\xi \models_{\mathbf{t}} \psi \,\mathtt{U}\, \theta$ implies $\xi \models_{\langle k,l+m\rangle} \psi \,\mathtt{U}\, \theta$. By (1), $\xi \models_{\langle k,l\rangle} \psi \,\mathtt{U}\, \theta$. Furthermore, $\mathbf{r} < \langle k+1,0\rangle$, so $\xi \models_{\langle k,l\rangle} \psi \,\mathtt{u}\, \theta$. Since $\psi \,\mathtt{u}\, \theta \in \mathrm{Sub}^*(\phi)$, similarly as above a contradiction follows from (4).

In the fourth case, in ξ, ψ holds in each world between $\mathbf{t}$ and $\mathbf{r}$. By conditions (2) and (3) the same holds for η. □

Theorem 3.2.4 establishes decidability in the following way: if ϕ is any formula and $M = (2\,\mathrm{lenght}(\phi) + 1)\, 4^{\mathrm{lenght}(\phi)}$, then ϕ is satisfiable iff there exists a function

$$\zeta : \{0,\ldots,M\}^2 \times Var(\phi) \longrightarrow \{0,1\}$$

such that $\zeta \models_{\langle 0,0\rangle} \phi$. Since ζ is not a model, the technical difficulty lies in the proper utilization of lemmas 3.2.8, 3.2.9 and 3.2.10 so that the initial model ξ such that $\xi \models_{\mathbf{t}} \phi$ can be transformed to the ultimately periodical model η with the following properties:

- $\eta \models_{\langle 0,0\rangle} \phi$;

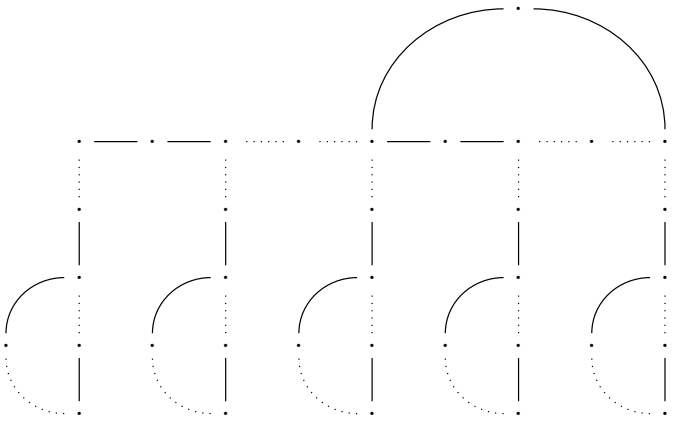


Figure 3.6: Ultimately periodic model

- $\mathrm{Sub}^*(\phi, \mathbf{r}, \eta)$ does not depend on η-values on $\{M, M+1, \dots\}^2 \times Var$ for all $\mathbf{r} \in \{0, \dots, M\}^2$.

Figure 3.6 illustrates the form of an ultimately periodic model.

Theorem 3.2.4 (Periodicity)**.** *Let $\xi \models_{\mathbf{t}} \phi$. Then, there exist a model η and positive integers k, m, l_i and n_i, $i = 1, \dots, k+m$ with the following properties:*

1. $\eta \models_{\langle 0,0 \rangle} \phi$;
2. $\eta(i,j,p) = \eta(i+m,j,p)$ *for all* $i > k$ *and all* $j \in \omega$;
3. $\eta(i,j,p) = \eta(i,j+n_i,p)$ *for all* $i < k+m$ *and all* $j \geqslant l_i$;
4. $\max(m, n_1, \cdots, n_{k+m}) \leqslant (2\,\mathrm{length}(\phi)+1) \cdot 4^{\mathrm{length}(\phi)}$;
5. $\max(k, l_1, \dots, l_{k+m}) \leqslant 4^{\mathrm{length}(\phi)}$.

Proof. Starting with a model ξ, the desired model η will be built gradually. Let

$$\eta_0(i,j,p) = \xi(i+t_1, j+t_2, p), \quad i,j \in \omega,\ p \in Var.$$

Clearly, $\eta_0 \models_{\langle i,j \rangle} \psi$ iff $\xi \models_{\langle i+t_1, j+t_2 \rangle} \psi$ for all $\psi \in For$, hence η_0 satisfies (1).

Recall that an equivalence relation $\sim$ on $\omega \times \omega$ has been defined by

$$\mathbf{r} \sim \mathbf{s} \text{ iff } \mathrm{Sub}^*(\phi, \mathbf{r}, \eta_0) = \mathrm{Sub}^*(\phi, \mathbf{s}, \eta_0).$$

Note that $\sim$ has finitely many equivalence classes. The number of classes is bounded by $4^{\mathrm{length}(\phi)}$, since there are at most $2\,\mathrm{length}(\phi)$ formulas in $\mathrm{Sub}^*(\phi)$. Consequently, at least one of the sets

$$A_i = \{\langle j, 0 \rangle \ :\ j \in \omega \text{ and } \langle i, 0 \rangle \sim \langle j, 0 \rangle\},\ i \in \omega$$

is infinite. Let i_0 be the smallest nonnegative integer such that A_{i_0} is infinite. Let $\psi_1 \,\mathtt{U}\, \theta_1, \dots, \psi_s \,\mathtt{U}\, \theta_s$ be the sequence of all $\mathtt{U}$-formulas from $\mathrm{Sub}^*(\phi, \langle i_0, 0 \rangle, \eta_0)$ and $\mathbf{r}_1, \dots, \mathbf{r}_s$ the sequence of time instances so that, for all $j \in \{1, \dots, s\}$, $\eta_0 \models_{\mathbf{r}_j} \theta_j$ and $\eta_0 \models_{\mathbf{t}} \psi_j$ for all $\mathbf{t} \in [\langle i_0, 0 \rangle, \mathbf{r}_j)_{\mathrm{lex}}$. Let j_0 be the smallest positive

integer such that $\langle i_0, 0\rangle \sim \langle j_0, 0\rangle$ and $\langle j_0, 0\rangle > \mathbf{r}_j$ for all $j \in \{1, \ldots, s\}$. It is easy to see that η_0, $\langle i_0, 0\rangle$ and $\langle j_0, 0\rangle$ satisfy conditions (1) and (4) of Lemma 3.2.9.

Without loss of generality it is possible to assume that $\mathbf{r}_1 \leqslant \cdots \leqslant \mathbf{r}_s$. Let $\mathbf{r}_j = \langle r_{j,1}, r_{j,2}\rangle$ for all $j \in \{1, \ldots, s\}$, and let $I_0 = (\langle r_{0,1}, 0\rangle, \langle r_{1,1}, 0\rangle)_{\text{lex}}$, $I_1 = (\langle r_{1,1}, 0\rangle, \langle r_{2,1}, 0\rangle)_{\text{lex}}$, $\ldots$, $I_s = (\langle r_{s,1}, 0\rangle, \langle r_{s+1,1}, 0\rangle)_{\text{lex}}$, where $r_{0,1} = i_0$ and $r_{s+1,1} = j_0$. Now, using the pseudo-code, the construction of the model η_1 and the positive integer $m \leqslant 2\,\text{lenght}(\phi)\, 4^{\text{length}(\phi)}$ will be described so that $\eta_1 \models_{\langle 0,0\rangle} \phi$ and η_1, $\langle i_0, 0\rangle$, $\langle i_0 + m, 0\rangle$ satisfy conditions (1) and (4) of Lemma 3.2.9:

$\eta' := \eta_0$;

For $j = 0$ to s do:

While there exist $\langle a, 0\rangle, \langle b, 0\rangle \in I_j$ so that $a < b$ and $\text{Sub}^*(\phi, \langle a, 0\rangle, \eta') = \text{Sub}^*(\phi, \langle b, 0\rangle, \eta')$ do:

$\eta''(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' < \langle a, 0\rangle$ and all $p \in Var$;
$\eta''(\langle a+i, i'\rangle, p) := \eta'(\langle b+i, i'\rangle, p)$ for all $i, i' \in \omega$ and all $p \in Var$;
$r_{l,1} := r_{l,1} + a - b$ for all $l \in \{j, \ldots, s+1\}$;
$I_l := (\langle r_{l,1}, 0\rangle, \langle r_{l+1,1}, 0\rangle)_{\text{lex}}$, for all $l \in \{j, \ldots, s\}$;
$\eta' := \eta''$;

$m := r_{s+1,1} - i_0$;

$\eta_1 := \eta'$.

Note that η'' is constructed from η' in such a way that η', η'' and ϕ satisfy conditions of Lemma 3.2.8. Furthermore, η' is initialized to be η_0, so by Lemma 3.2.8, $\eta_1 \models_{\langle 0,0\rangle} \phi$. Moreover, the immediate consequence of the construction of η_1 is the fact that $\text{Sub}^*(\phi, \langle i_0, 0\rangle, \eta_1) = \text{Sub}^*(\phi, \langle i_0 + m, 0\rangle, \eta_1)$ and that η_1, $\langle i_0, 0\rangle$ and $\langle i_0 + m, 0\rangle$ satisfy condition (4) of Lemma 3.2.9. Note that for all $j \in \{0, \ldots, s\}$, and all $\langle a, 0\rangle, \langle b, 0\rangle \in I_j$ by the construction we have that

$$\langle a, 0\rangle \not\sim_{\phi,\eta_1} \langle b, 0\rangle.$$

Since $s \leqslant 2\,\text{length}(\phi)+1$ and the number of classes of $\sim$ is bounded by $4^{\text{length}(\phi)}$, $m \leqslant (2\,\text{lenght}(\phi) + 1)\, 4^{\text{lenght}(\phi)}$ holds.

As the next step towards the desired model η the model η_2 and the positive integer $k \leqslant 4^{\text{length}(\phi)}$ will be constructed so that $\eta_2 \models_{\langle 0,0\rangle} \phi$ and η_2, $\langle k, 0\rangle$, $\langle k + m, 0\rangle$ satisfy conditions (1) and (4) of Lemma 3.2.9. As above, the pseudo-code will be used to describe the construction of η_2 and k:

$\eta' := \eta_1$;

$k := i_0$;

While there exist $a < b \leqslant k$ so that $\text{Sub}^*(\phi, \langle a, 0\rangle, \eta') = \text{Sub}^*(\phi, \langle b, 0\rangle, \eta')$ do:

$\eta''(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' < \langle a, 0\rangle$ and all $p \in Var$;
$\eta''(\langle a + i, i'\rangle, p) := \eta'(\langle b + i, i'\rangle, p)$ for all $i, i' \in \omega$ and all $p \in Var$;

$k := k + a - b$;

$\eta' := \eta''$;

$\eta_2 := \eta'$.

Similarly as in the case of η_1, it is possible to verify that η_2 and k satisfy conditions (1) and (5) of the present theorem.

Now, the construction of the model η_3 and positive integers l_i, n_i, $i \in \{1, \ldots, k+m\}$ will be described so that conditions (1), (3)-(5) of the present theorem are satisfied.

$\eta' := \eta_2$;

For $i = 0$ to $k + m - 1$ do:

Find positive integers $j_{i,1}, j_{i,2}$ so that $j_{i,1} < j_{i,2}$, $\langle i, j_{i,1}\rangle \sim_{\phi,\eta'} \langle i, j_{i,2}\rangle$ and η', $\langle i, j_{i,1}\rangle$, $\langle i, j_{i,2}\rangle$ satisfy condition (4) of Lemma 3.2.10. The existence of such $j_{i,1}$ and $j_{i,2}$ can be proved similarly as it has been done in the case of η_0, $\langle i_0, 0\rangle$ and $\langle j_0, 0\rangle$, prior to the construction of η_1;

Form the set $\{\psi_1 \,\mathtt{u}\, \theta_1, \ldots, \psi_v \,\mathtt{u}\, \theta_v\} \subseteq \mathrm{Sub}^*(\phi, \langle i, j_{i,1}\rangle, \eta')$ of all $\mathtt{u}$-formulas in $\mathrm{Sub}^*(\phi, \langle i, j_{i,1}\rangle, \eta')$;

For each $j \in \{1, \ldots, v\}$ find the smallest nonnegative integer r_j such that $j_{i,1} \leqslant r_j < j_{i,2}$, $\eta' \models_{\langle i, r_j\rangle} \theta_j$ and for all $\mathtt{t} \in [\langle i, j_{i,1}\rangle, \langle i, r_j\rangle)_{\mathrm{lex}}$, $\eta' \models_{\mathtt{t}} \psi_j$ holds. Without loss of generality it is possible to assume that $\mathtt{r}_1 \leqslant \cdots \leqslant \mathtt{r}_v$;

$r_0 := j_{i,1}$, $r_{v+1} := j_{i,2}$;

$I_0 := (\langle i, r_0\rangle, \langle i, r_1\rangle)_{\mathrm{lex}}$, $\ldots$, $I_v := (\langle i, r_v\rangle, \langle i, r_{v+1}\rangle)_{\mathrm{lex}}$;

For $j = 0$ to v do:

While there exist $\langle i, a\rangle, \langle i, b\rangle \in I_j$ so that $a < b$ and

$$\mathrm{Sub}^*(\phi, \langle i, a\rangle, \eta') = \mathrm{Sub}^*(\phi, \langle i, b\rangle, \eta')$$

do:

$\eta''(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' < \langle i, a\rangle$ and all $p \in Var$;

$\eta''(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' \geqslant \langle i+1, 0\rangle$ and all $p \in Var$;

$\eta''(\langle i, a + i'\rangle, p) := \eta'(\langle i, b + i'\rangle, p)$ for all $i' \in \omega$ and all $p \in Var$;

$r_l := r_l + a - b$ for all $l \in \{j, \ldots, v+1\}$;

$I_l := (\langle i, r_l\rangle, \langle i, r_{l+1}\rangle)_{\mathrm{lex}}$ for all $l \in \{j, \ldots, v+1\}$;

$\eta' := \eta''$;

$n_{i+1} := r_{v+1} - r_0$;

$l_{i+1} := r_0$;

While there exist $a < b \leqslant l_{i+1}$ so that

$$\mathrm{Sub}^*(\phi, \langle i, a\rangle, \eta') = \mathrm{Sub}^*(\phi, \langle i, b\rangle, \eta')$$

do:

$\eta''(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' < \langle i, a\rangle$ and all $p \in Var$;
$\eta''(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' \geqslant \langle i+1, 0\rangle$ and all $p \in Var$;
$\eta''(\langle i, a+i'\rangle, p) := \eta'(\langle i, b+i'\rangle, p)$ for all $i' \in \omega$ and all $p \in Var$;
$l_{i+1} := l_{i+1} + a - b$;
$\eta' := \eta''$;

$\eta_3(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' < \langle i, l_{i+1} + n_{i+1}\rangle$ and all $p \in Var$;
$\eta_3(\mathbf{t}', p) := \eta'(\mathbf{t}', p)$ for all $\mathbf{t}' \geqslant \langle i+1, 0\rangle$ and all $p \in Var$;
$\eta_3(\langle i, d + n_{i+1}\rangle, p) := \eta_3(\langle i, d\rangle, p)$ for all $d \geqslant l_{i+1}$.

Using the similar argument as above it is possible to show that constructed integers l_i and n_j are bounded by $4^{\text{length}(\phi)}$ and $(2\,\text{length}(\phi)+1)\,4^{\text{length}(\phi)}$ respectively. The third condition follows immediately from the definition of η_3, while the first condition is preserved by Lemmas 3.2.10 and 3.2.8.

Finally, the model η_3 has to be transformed to the desired model η so that the second condition of the theorem is also satisfied. Since it has been already determined the beginning of the outer loop k and the outer period m, the final model η is defined as follows:

$\eta(\mathbf{t}', p) = \eta_3(\mathbf{t}', p)$ for all $\mathbf{t}' < \langle k+m, 0\rangle$ and all $p \in Var$;

$\eta(\langle i+k+m, j\rangle = \eta(\langle i+k, j\rangle))$ for all $i, j \in \omega$ and all $p \in Var$.

By construction,

$$\langle k, 0\rangle \sim_{\phi,\eta_3} \langle k+m, 0\rangle$$

and each $\mathtt{U}$-formula from the set $\text{Sub}^*(\phi, \langle k, 0\rangle, \eta_3)$ is fulfilled before $\langle k+m, 0\rangle$, so by Lemma 3.2.9, the model η satisfies conditions (1) and (2) and hence all conditions of the theorem. □ □

Theorem 3.2.5. *The satisfiability of $L([1], [\omega], \mathtt{u}, \mathtt{U})$-formulas is PSPACE complete.*

Proof. On the one hand, in [75] it is shown that the satisfiability of $L([1], \mathtt{u})$-formulas is PSPACE complete, so the lower bound of satisfiability of $L([1], [\omega], \mathtt{u}, \mathtt{U})$-formulas is PSPACE. For the other direction, a nondeterministic Turing machine that determines satisfiability and uses polynomial space with respect to the length of a given formula will be constructed. Before the start of the actual description of the TM, the following the notation and terminology will be used:

- k, m: nonnegative integers that have the same meaning as the corresponding numbers in Theorem 3.2.4 (k is the size of the initial segment, while m is the size of the outer loop);
- k_{local}, m_{local}: local versions of k and m (they are locally restricted to the current inner loop $\{\langle i, n\rangle \mid n \in \omega\}$);
- S_{start}: formulas from $\text{Sub}^*(\phi)$ guessed to be true at $\langle 0, 0\rangle$;
- S_{present}: formulas from $\text{Sub}^*(\phi)$ guessed to be true at the present moment;
- $S_{[1]}$: formulas from $\text{Sub}^*(\phi)$ guessed to be true at the next time instant $\langle i, j+1\rangle$;

- $S_{[\omega]}$: formulas from $\mathrm{Sub}^*(\phi)$ guessed to be true at ω-jump $\langle i+1, 0\rangle$;
- S_{in}: formulas from $\mathrm{Sub}^*(\phi)$ guessed to be true at the beginning of the inner period $\langle i, k_{\mathrm{loc}}\rangle$;
- $S_{\mathtt{u}}$: the set of all $\mathtt{u}$-formulas from S_{in};
- S_{out}: formulas from $\mathrm{Sub}^*(\phi)$ guessed at the beginning of the outer period $\langle k, 0\rangle$;
- $S_{\mathtt{U}}$: the set of all $\mathtt{U}$-formulas from S_{out};
- Any of sets S_* is said to satisfy *Boolean consistency* iff for all $\psi, \theta \in Sub(\phi)$ the following is true:
  - $\psi \in S_*$ or $\neg\psi \in S_*$;
  - $\psi \in S_* \Leftrightarrow \neg\psi \notin S_*$;
  - $\psi \wedge \theta \in S_* \Leftrightarrow \psi, \theta \in S_*$;
- S_{present}, $S_{[1]}$ and $S_{[\omega]}$ are *properly linked* iff for all $\psi, \theta \in Sub(\phi)$ the following conditions are satisfied:
  - $[1]\psi \in S_{\mathrm{present}} \Leftrightarrow \psi \in S_{[1]}$;
  - $[\omega]\psi \in S_{\mathrm{present}} \Leftrightarrow \psi \in S_{[\omega]}$;
  - $\psi\,\mathtt{u}\,\theta \in S_{\mathrm{present}}$ iff $\theta \in S_{\mathrm{present}}$, or $\psi, \neg\theta \in S_{\mathrm{present}}$ and $\psi\,\mathtt{u}\,\theta \in S_{[1]}$. Here $\neg\theta \in S_{\mathrm{present}}$ has the same meaning as $\theta \notin S_{\mathrm{present}}$;
  - If the counter is not on the end of the inner loop, then $\psi\,\mathtt{U}\,\theta \in S_{\mathrm{present}}$ iff $\theta \in S_{\mathrm{present}}$, or $\psi, \neg\theta \in S_{\mathrm{present}}$ and $\psi\,\mathtt{U}\,\theta \in S_{[1]}$. If the counter is on the end of the inner loop, then $\psi\,\mathtt{U}\,\theta \in S_{\mathrm{present}}$ iff $\theta \in S_{\mathrm{present}}$, or $\psi, \neg\theta \in S_{\mathrm{present}}$ and $\psi\,\mathtt{U}\,\theta \in S_{[\omega]}$.

TM, that determines satisfiability, works as follows:

input ϕ;

guess k, m, S_{start} and S_{out};

check Boolean consistency of S_{start} and S_{out}; if it fails return false;

if $\phi \notin S_{\mathrm{start}}$ return false;

construct $S_{\mathtt{U}}$;

$S_{\mathrm{present}} := S_{\mathrm{start}}$;

for $i = 0$ to $k + m - 1$ do;

guess k_{loc}, m_{loc}, S_{in};
if $i < k + m - 1$ guess $S_{[\omega]}$; else $S_{[\omega]} := S_{\mathrm{out}}$;
check Boolean consistency of S_{in} and $S_{[\omega]}$; if it fails return false;
construct $S_{\mathtt{u}}$;
for $j = 0$ to $k_{\mathrm{loc}} + m_{\mathrm{loc}} - 1$ do;

if $j < k_{\text{loc}} + m_{\text{loc}} - 1$ guess $S_{[1]}$; else $S_{[1]} := S_{\text{in}}$;
check Boolean consistency of $S_{[1]}$; if it fails return false;
check whether S_{present}, $S_{[1]}$ and $S_{[\omega]}$ are properly linked; if it fails return false;
if $j \geqslant k_{\text{loc}}$ then for all $\psi \,\mathtt{u}\, \theta \in S_{\mathtt{u}}$ check whether $\theta \in S_{\text{present}}$; if it passes, delete $\psi \,\mathtt{u}\, \theta$ from $S_{\mathtt{u}}$;
if $k \leqslant i < k + m - 1$ then for all $\psi \,\mathtt{U}\, \theta \in S_{\mathtt{U}}$ check whether $\theta \in S_{\text{present}}$; if it passes, delete $\psi \,\mathtt{U}\, \theta$ from $S_{\mathtt{U}}$;
$S_{\text{present}} := S_{[1]}$;
next j;

if $S_{\mathtt{u}} \neq \emptyset$ return false;

$S_{\text{present}} := S_{[\omega]}$;

next i;

if $S_{\mathtt{U}} \neq \emptyset$ return false;

end.

It is easy to see that the TM just described uses polynomial space with respect to length(ϕ), so the satisfiability problem is at most PSPACE hard. Since PSPACE is both upper and lower complexity bound, the theorem holds. □

3.2.4 Related work

The motivation for this particular modification of discrete linear time temporal logic has come from the research of A. Gargantini, D. Mandrioli and A. Morzenti that was presented in [44].

In the recent companion paper [41], authors use concepts from non-standard analysis and provide notions of micro and macro steps in an extension of the TRIO metric temporal general-purpose specification language. The key difference between that paper and the approach described here is that here the time flow is restricted to a concrete well-ordering.

In [26, 27] a family of temporal logics that extend LTL to allow time flows isomorphic to any countable limit ordinal are presented. Decidability of those logics is analyzed using generalized Büchi automata. The logic described there corresponds to the logic with time isomorphic to ω^2, where the considered operators (in their notation) are $\bigcirc^1$, $\bigcirc^\omega$ and $\mathtt{U}^{\omega^2}$.

The results presented in this section can be classified as a research related to discrete linear time temporal logics, with particular application on system descriptions and handling zero-time transitions in Petri nets. For modal and temporal part, reader can refer to [1, 9, 17–20, 33–35, 40, 43, 48, 55, 61, 68].

The infinitary techniques presented here (application of infinitary inference rules in order to overcome inherited noncompactness) are connected with the research presented in [30, 31, 67].

Decidability argumentation presented here is a modification of the work of A. Sistla and E. Clarke presented in [75].

3.3 Refinement proof of ASM Chord verification

In this Section the proof of Theorems 3.1.2 – 3.1.5 will be given assuming the refined format that the time for execution of a step which is a part of a rule is infinitely smaller than the time used for the communication between two nodes, and using low level definition of the Chord rules given in Appendix A. The new set of allowed executions, called ε-regular runs, is introduced as a special kind of *regular runs*:

Definition 3.3.1 (ε-regular runs)**.** *ε-regular runs are regular runs that satisfy:*

- $\neg fair\_leave(n)\,\mathtt{U}\,member\_of\_stable\_pair(n)$, and
- $\neg unfair\_leave(n)\,\mathtt{U}\,member\_of\_stable\_pair(n)$, and
- $\neg put(n)\,\mathtt{U}\,member\_of\_stable\_pair(n)$, and
- $\mathtt{F}\,execute\_rule(n)$

for every node $n \in Node$, where $member\_of\_stable\_pair(n)$, $fair\_leave(n)$, $unfair\_leave(n)$, $put(n)$ and $execute\_rule(n)$ denote respectively a relation that n is a member of a stable pair of nodes, n can execute FairLeave, UnfairLeave, Put *or its next rule.*

Note that, for the finite set $P = \{p_1, \ldots, p_k\}$ and the given unary predicate $R \subset P$, it is possible to introduce new propositional letters $pr_1, \ldots, pr_k$ such that $R(p_i)$ holds *iff* $pr_i = \top$, i.e. it is possible to code $R(x)$ into the propositional case.

Theorem 3.3.1. *Let a peer join a Chord network, between two nodes which constitute a stable pair. Then, there is a number $k > 0$ of steps, such that if no other join rule happens in the meantime, the* Stabilize *rule will bring the starting pair to be stable after k steps.*

Proof. Suppose that the network contains only one node N_1 and that N_2 wants to join. The following sequence of moves will be considered:

State	Rule	Node	Values
$S_{\langle 0,0\rangle}$			$successor(N_1) = N_1$ $predecessor(N_1) = N_1$
$S_{\langle 1,0\rangle}$	JoinBegin	New	
$S_{\langle 1,1\rangle}$			$hash(New) = N_2$
$S_{\langle 1,2\rangle}$			$predecessor(N_2) = undef$
$S_{\langle 2,0\rangle}$	JoinGetSuccessor	N_2	
$S_{\langle 2,1\rangle}$			$\boldsymbol{successor(N_2) = N_1}$
$S_{\langle 3,0\rangle}$	StabilizeBegin	N_2	
$S_{\langle 4,0\rangle}$	StabilizeWithPredecessor	N_2	
$S_{\langle 4,1\rangle}$			$x = N_1$
$S_{\langle 5,0\rangle}$	ReadMessage	N_1	
$S_{\langle 5,1\rangle}$			$\boldsymbol{predecessor(N_1) = N_2}$
$S_{\langle 6,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 7,0\rangle}$	StabilizeWithPredecessor	N_1	

$S_{\langle 7,1\rangle}$			$x = N_2$
$S_{\langle 7,2\rangle}$			$\boldsymbol{successor(N_1) = N_2}$
$S_{\langle 8,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 9,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 9,1\rangle}$			$x = N_2$
$S_{\langle 10,0\rangle}$	ReadMessage	N_1	
$S_{\langle 10,1\rangle}$			$\boldsymbol{predecessor(N_2) = N_1}$

Obviously, a stable pair has been established, again.

Suppose that there are two or more nodes in the network, and that N_2 wants to join. Let N_1 and N_3 be the members of the network such that $successor(1) = 3$, and $predecessor(3) = 1$ (i.e., $\langle N_1, N_3\rangle$ is a stable pair).

State	Rule	Node	Values
$S_{\langle 0,0\rangle}$			$successor(N_1) = N_3$ $predecessor(N_3) = N_1$
$S_{\langle 1,0\rangle}$	JoinBegin	New	
$S_{\langle 1,1\rangle}$			$hash(New) = N_2$
$S_{\langle 1,2\rangle}$			$predecessor(N_2) = undef$
$S_{\langle 2,0\rangle}$	JoinGetSuccessor	N_2	
$S_{\langle 2,1\rangle}$			$\boldsymbol{successor(N_2) = N_3}$
$S_{\langle 3,0\rangle}$	StabilizeBegin	N_2	
$S_{\langle 4,0\rangle}$	StabilizeWithPredecessor	N_2	
$S_{\langle 4,1\rangle}$			$x = N_1$
$S_{\langle 5,0\rangle}$	ReadMessage	N_3	
$S_{\langle 5,1\rangle}$			$\boldsymbol{predecessor(N_3) = N_2}$
$S_{\langle 6,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 7,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 7,1\rangle}$			$x = N_2$
$S_{\langle 7,2\rangle}$			$\boldsymbol{successor(N_1) = N_2}$
$S_{\langle 8,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 9,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 9,1\rangle}$			$x = N_2$
$S_{\langle 10,0\rangle}$	ReadMessage	N_1	
$S_{\langle 10,1\rangle}$			$\boldsymbol{predecessor(N_2) = N_1}$

Thus, a stable pair has been established. □

Theorem 3.3.2 (Concurrent joins)**.** *Let a Chord network contain a stable pair. If a sequence of* Join *rules is executed between the nodes which form this stable pair, interleaved with* Stabilize, UpdatePredecessor *and* Update_fingers*, then there is a number $k > 0$ of steps, such that after the last* Join *rule, the starting pair of nodes will be stable after k steps.*

Proof. First, note that UpdateFingers does not change the values of the functions *predecessor* and *successor*. Similarly, UpdatePredecessor might change values of the function *predecessor* only after an UnfairLeave. Thus, executions of UpdatePredecessor and UpdateFingers will not be considered in the rest of this proof.

If it is assumed that all peers that want to join the network have different successors. Then, by Theorem 3.3.1, the statement holds. Otherwise, there must be at least two peers that want to join the network having the same successor. Suppose that N_2 and N_3 want to join and that N_1 and N_4 are members of the network, such that $successor(1) = 4$ and $predecessor(4) = 1$. Then, the following sequence of moves will be considered:

State	**Rule**	**Node**	**Values**
$S_{\langle 0,0\rangle}$			$successor(N_1) = N_4$ $predecessor(N_4) = N_1$
$S_{\langle 1,0\rangle}$	JoinBegin	New_1	
$S_{\langle 1,1\rangle}$			$hash(New_1) = N_2$
$S_{\langle 1,2\rangle}$			$predecessor(N_2) = undef$
$S_{\langle 2,0\rangle}$	JoinBegin	New_2	
$S_{\langle 2,1\rangle}$			$hash(New_2) = N_3$
$S_{\langle 2,2\rangle}$			$predecessor(N_3) = undef$
$S_{\langle 3,0\rangle}$	JoinGetSuccessor	N_2	
$S_{\langle 3,1\rangle}$			$successor(N_2) = N_4$
$S_{\langle 4,0\rangle}$	JoinGetSuccessor	N_3	
$S_{\langle 4,1\rangle}$			$\boldsymbol{successor(N_3) = N_4}$
$S_{\langle 5,0\rangle}$	StabilizeBegin	N_3	
$S_{\langle 6,0\rangle}$	StabilizeWithPredecessor	N_3	
$S_{\langle 6,1\rangle}$			$x = N_1$
$S_{\langle 7,0\rangle}$	ReadMessage	N_4	
$S_{\langle 7,1\rangle}$			$\boldsymbol{predecessor(N_4) = N_3}$
$S_{\langle 8,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 9,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 9,1\rangle}$			$x = N_3$
$S_{\langle 9,2\rangle}$			$successor(N_1) = N_3$
$S_{\langle 10,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 11,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 11,1\rangle}$			$x = N_3$
$S_{\langle 12,0\rangle}$	ReadMessage	N_3	
$S_{\langle 12,1\rangle}$			$predecessor(N_3) = N_1$
$S_{\langle 13,0\rangle}$	StabilizeBegin	N_2	
$S_{\langle 14,0\rangle}$	StabilizeWithPredecessor	N_2	
$S_{\langle 14,1\rangle}$			$x = N_3$
$S_{\langle 14,2\rangle}$			$\boldsymbol{successor(N_2) = N_3}$
$S_{\langle 15,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 16,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 16,1\rangle}$			$x = N_2$
$S_{\langle 17,0\rangle}$	ReadMessage	N_3	
$S_{\langle 17,1\rangle}$			$\boldsymbol{predecessor(N_3) = N_2}$

$S_{\langle 18,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 19,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 19,1\rangle}$			$x = N_2$
$S_{\langle 19,2\rangle}$			$\boldsymbol{successor(N_1) = N_3}$
$S_{\langle 20,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 21,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 21,1\rangle}$			$x = N_2$
$S_{\langle 22,0\rangle}$	ReadMessage	N_2	
$S_{\langle 22,1\rangle}$			$\boldsymbol{predecessor(N_2) = N_1}$

Thus, a stable pair has been established. □

Theorem 3.3.3. *Let a Chord network contain a stable pair and let a node between them leave the network. Then, there is a number $k \geqslant 0$ of steps, such that if no* Join *rule happens at the considered part of the network in the meantime.*

Proof. If it is assumed that the node leaves the network in a fair way, since FairLeave produces a stable pair, the statement holds for $k = 0$. Thus, let UnfairLeave be executed.

Suppose that the network contains only two nodes N_1 and N_2, and that N_2 leaves in an unfair way and breaks the ring. Then, the following sequence of moves will be considered:

State	Rule	Node	Values
$S_{\langle 0,0\rangle}$			$successor(N_1) = N_2$ $predecessor(N_1) = N_2$ $successor(N_2) = N_1$ $predecessor(N_2) = N_1$
$S_{\langle 1,0\rangle}$	UnfairLeave	N_2	
$S_{\langle 2,0\rangle}$	UpdatePredecessor	N_1	
$S_{\langle 2,1\rangle}$			$predecessor(N_1) = undef$
$S_{\langle 3,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 4,0\rangle}$	StabilizeWaitSuccessor	N_1	
$S_{\langle 4,1\rangle}$			$\boldsymbol{successor(N_1) = N_1}$
$S_{\langle 5,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 6,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 6,1\rangle}$			$x = undef$
$S_{\langle 7,0\rangle}$	ReadMessage	N_1	
$S_{\langle 7,1\rangle}$			$\boldsymbol{predecessor(N_1) = N_1}$

Suppose that there are three or more nodes in a network. Let N_1, N_2 and N_3 be the members of the network such that $successor(1) = 2$ and $successor(2) = 3$. Suppose that N_2 unfair leaves and breaks the ring of the successors pointers. Then, the following sequence of moves which results with the stable pair $\langle N_1, N_3\rangle$ will be considered:

State	Rule	Node	Values
$S_{\langle 0,0\rangle}$			$successor(N_1) = N_2$ $predecessor(N_2) = N_1$ $successor(N_2) = N_3$ $predecessor(N_3) = N_2$
$S_{\langle 1,0\rangle}$	UnfairLeave	N_2	
$S_{\langle 2,0\rangle}$	UpdatePredecessor	N_3	
$S_{\langle 2,1\rangle}$			$predecessor(N_3) = undef$
$S_{\langle 3,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 4,0\rangle}$	StabilizeWaitSuccessor	N_1	
$S_{\langle 4,1\rangle}$			$\boldsymbol{successor(N_1) = N_3}$
$S_{\langle 5,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 6,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 6,1\rangle}$			$x = undef$
$S_{\langle 7,0\rangle}$	ReadMessage	N_3	
$S_{\langle 7,1\rangle}$			$\boldsymbol{predecessor(N_3) = N_1}$

□

Theorem 3.3.4. *Let a Chord network contain a stable pair. Let a node which is between those nodes leave the network following by several nodes which want to join between them. Then, there is a number $k \geqslant 0$ of steps, such that the considered pair will be brought into a stable state after k steps.*

Proof. If it is assumed that the node leaves the network in a fair way. It produces a stable pair, and according to the theorems 3.3.2 and 3.3.3, the statement holds. Then, let the node N_2 execute UnfairLeave and break the ring. If no node joins the network in the ring interval $[predecessor(2), successor(2)]$, the statement holds similarly as in Theorems 3.3.2 and 3.3.3. Finally, assume that a node joins the network in the ring interval $[predecessor(2), successor(2)]$. Suppose that N_1, N_2 and N_4 are members of the network, such that $successor(1) = 2$, $predecessor(2) = 1$, $successor(2) = 4$ and $predecessor(4) = 2$. Let N_2 be the node that will leave, and N_3 node that will join the network. The following sequence of moves will be considered:

State	Rule	Node	Values
$S_{\langle 0,0\rangle}$			$successor(N_1) = N_2$ $predecessor(N_2) = N_1$ $successor(N_2) = N_4$ $predecessor(N_4) = N_2$
$S_{\langle 1,0\rangle}$	UnfairLeave	N_2	
$S_{\langle 2,0\rangle}$	JoinBegin	New	
$S_{\langle 2,1\rangle}$			$hash(New) = N_3$
$S_{\langle 2,2\rangle}$			$predecessor(N_3) = undef$
$S_{\langle 3,0\rangle}$	JoinGetSuccessor	N_3	
$S_{\langle 3,1\rangle}$			$\boldsymbol{successor(N_3) = N_4}$

$S_{\langle 4,0\rangle}$	UpdatePredecessor	N_4	
$S_{\langle 4,1\rangle}$			$predecessor(N_4) = undef$
$S_{\langle 5,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 6,0\rangle}$	StabilizeWaitSuccessor	N_1	
$S_{\langle 6,1\rangle}$			$successor(N_1) = N_4$
$S_{\langle 7,0\rangle}$	StabilizeBegin	N_3	
$S_{\langle 8,0\rangle}$	StabilizeWithPredecessor	N_3	
$S_{\langle 8,1\rangle}$			$x = undef$
$S_{\langle 9,0\rangle}$	ReadMessage	N_4	
$S_{\langle 9,1\rangle}$			$\boldsymbol{predecessor(N_4) = N_3}$
$S_{\langle 10,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 11,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 11,1\rangle}$			$x = N_3$
$S_{\langle 11,2\rangle}$			$\boldsymbol{successor(N_1) = N_3}$
$S_{\langle 12,0\rangle}$	StabilizeBegin	N_1	
$S_{\langle 13,0\rangle}$	StabilizeWithPredecessor	N_1	
$S_{\langle 13,1\rangle}$			$x = N_3$
$S_{\langle 14,0\rangle}$	ReadMessage	N_3	
$S_{\langle 14,1\rangle}$			$\boldsymbol{predecessor(N_3) = N_1}$

which results with the stable starting pair. If more than one node want to join the network in the considered ring interval, the similar arguments as in the proof of Theorem 3.3.2 can be used to establish the statement. □

4

Synapse Protocol

As it is said in the Introduction, one of the possible solutions for the interconnection of heterogeneous overlay networks is the Synapse protocol. It was introduced in [59]. It is a generic and flexible meta-protocol that provides simple mechanisms and algorithms for easy interconnection of overlay networks. The Synapse protocol has already found some applications in the real-world situations that is presented in [24, 64].

Some of the results presented in this chapter are the result of work by the other members of the INRIA LogNet team (`http://www-sop.inria.fr/teams/lognet/LOGNET/`), and co-authors of [59], but they will presented here to give better understanding of all the aspects of the Synapse protocol. Figures that are inherited from [59] and that are work of my co-authors are marked with * in the captions.

4.1 Definition

Information is a set of basic $\langle key, value \rangle$ pairs, as commonly encountered in protocols for information retrieval. The protocol specifies how to insert information (`PUT`), how to retrieve it through a key (`GET`) and how to join a given overlay (`JOIN`) over a heterogeneous collection of overlay networks linked by co-located nodes. These co-located nodes represent a simple way to aggregate the resources of distinct overlays. It is assumed that each overlay has its own inner routing algorithm, called by the Synapse protocol to route requests inside each overlay. Also, it is assumed that there is no knowledge of the logical topology of all the involved overlay networks connected by Synapse. To ensure the usual properties of the underlying network, it is assumed that communication is both symmetric and transitive. Synapse simply ignores about how routing takes place inside the overlays, Synapse only offers a mechanism to route from one overlay to another in a simple, scalable and efficient way.

The inter-overlay network, induced by the Synapse protocol, can be considered as an aggregation of heterogeneous sub-overlay networks (referred to as *intra*-overlay networks henceforth). Each intra-overlay consists of one instance of, e.g., Chord [78–80] or any structured, unstructured or hybrid overlay, equipped with a $\langle key, value \rangle$ distribution and retrieval mechanism. It is assumed that an overlay network for information retrieval consists of a set of nodes on which the

information on some resources is distributed. Each intra-overlay has its own hash function, $\langle key, value \rangle$ distribution and retrieval policy, logical topology, search complexity, routing and fault-tolerance mechanisms, so on and so forth. The Synapse protocol can be summarized by the following points:

- *Synapses:* the interconnection of intra-overlay networks is achieved by co-located nodes taking part in several of these intra-overlays, called synapses. Each peer will act according to the policy of each of its intra-overlays, but will have the extra-role of forwarding the requests to some intra-overlay it belongs to.

- *Peer's name:* every peer comes with a proper logical name in each intra-overlay; in particular, synapses have as many logical names as the number of networks they belongs to.

- *Keys mapping in peers:* each peer is responsible for a set of resources $\langle key, value \rangle$ it hosts. Since every intra-overlay has different policies for keys distribution, it is possible to say that also the inter-overlay induced by Synapse also inherits homogeneous distribution among the intra- and inter-networks. As for peers, every key comes with a proper logical name peculiar to each intra-overlay.

- *Set of resources assigned to set of nodes:* all overlay protocols for information retrieval share the invariant of having a set of peers responsible of a specific set of resources. This invariant allows for routing under structured, unstructured and hybrid networks, because by construction, intra-routing is the one always responsible for its correctness, since Synapse just cares about overlay's inter-connection.

- *Network independency and message translation:* intra-network protocols are different by construction: as such, when a message leaves a particular network and enters another network, the first network loses control of the route of that message inside the second one.

- *Topology, exhaustiveness, complexity and scalability:* by construction, the inter-overlay network induced by the Synapse protocol belongs to the category of unstructured overlay networks, with a routing that is not exhaustive, even if Synapse can connect only overlays that guarantee exhaustivity. The same goes for the routing complexity that can be upper-bounded only in the presence of precise and strong hypotheses about the type of intra-overlay networks. The same goes for scalability: a Synapse inter-overlay is scalable if all the intra-networks are scalable.

- *Loopy routing avoidance:* to avoid lookup cycles when doing inter-routing, each peer maintains a list of tags of already processed requests, in order to discard previously seen queries, and a TTL value, which is decreased at each hop. These two features prevent the system from generating loops and useless queries, thus reducing the global number of messages in the Synapse inter-overlay.

4.1.1 "White box" vs. "black box" Synapse protocol

One important issue in interconnecting overlay networks is the ability of one overlay to potentially modify its protocol instead of only accepting that co-located nodes will route packets without any change in the protocol itself. This is a concrete backward compatibility issue, since many overlays already exist, and it is hard to change them at this point for many reasons (security, commercial, technological ...).

As such, two models of the Synapse protocol have been developed. The first *white box* model, is suitable to interconnecting overlays whose standards are open and collaborative, meaning that the protocol and the software client can be modified accordingly. The second, *black box* model, is suitable to interconnecting overlays that, for different reasons, are not collaborative at all, in the sense that they only route packets according to their proprietary and immutable protocol. The *white box* allows the adding of extra parameters to the current inter-overlay that are connected, while the *black box* deals with those extra parameters by means of a *synapse control network*, i.e. a distributed overlay that stores all the synapse parameters that cannot be carried on by some intra-overlay network.

White box Synapse

The *white box* hereby presented is capable of connecting heterogeneous network topologies given the assumption that every node is aware of the additions made to existing overlay protocols. The new parameters used to handle the game over strategy and replication need to be embedded into the existing protocols, so does the unhashed key in order to be rehashed when a synapse is met. One important requirement of the Synapse *white box* protocol with respect to other protocols using hash functions is that the keys and nodes' addresses circulate *unhashed* from hop to hop. Hash functions have no inverse: once a sought key is hashed, it is impossible to retrieve its initial value, and thus impossible to forward to another overlay having a different hash function, since hash functions may vary (in implementations and keysize) from overlay to overlay. Both the hashed and the *clear* key data can be carried within the message, or a fast hash computation can be performed at each step. Standard cryptographic protocols can be used in case of strong confidentiality requirements, without affecting the scalability of the Synapse protocol itself.

The following pseudo code illustrates the instructions of the *white box* version of the Synapse protocol:

```
on receipt of OPE(code, key, value)                receive an operation code
   from ipsend do                                  a key and a value from ipsend
      tag = this.new_tag(ipsend);                  create a new unique tag for this lookup
      send FIND(code, ttl, mrr, tag, key, value, ipsend)   send a FIND
         to this.myip;                             message with code ... ipsend to itself

on receipt of                                      receive a FIND message with code ... ipdest from ipsend
   FIND(code, ttl, mrr, tag, key, value, ipdest) from ipsend do
   if ttl = 0                                      the lookup is aborted because of zero ttl or because
      or this.game_over?(tag)                      of the game over strategy
   else this.push_tag(tag);                        push the tag of the query as "already processed"
```

```
        next_mrrs = distrib_mrr(mrr, this.net_list);        fix the assoc list
splitting
                                          all the mrr between all net the synapse belongs to
        for all net ∈ this.net_list do      for all net the synapse belongs do
            if this.isresponsible?(net, key)   the current synapse is responsible
                                                          for the key in the net
                send FOUND(code, net, mrr, key, value)      send a FOUND
                    to ipdest;                  message with code ... value to ipdest

on receipt of FOUND(code, net, mrr, key, value)        receive a FOUND
    from ipsend do                          message with code ... value from ipsend
    match code
        code = GET                                                    GET code
            send READ_TABLE(net, key)           send a READ_TABLE
message
                to ipsend          (omitted) through the net with key to ipsend
        code = PUT                                                    PUT code
            if mrr < 0          stop replication since no inter PUT is allowed
            else send            send a WRITE_TABLE message (omitted)
                WRITE_TABLE(net, key, value)    through the net with key
                    to ipsend                          and value to ipsend

on receipt of JOIN(net)                    receive a JOIN message to reach
    from ipsend do                                        the net from ipsend
        this.insert_net(net, ipsend)  the current synapse insert ipsend in the net
```

Black box synapse

Interconnecting existing overlays made of "blind" peers, who are not aware of any additional parameters, seems to be a natural Synapse evolution and it constitutes a problem worth investigating. The assumption is that an overlay can be populated by blind peers (e.g. nodes previously in place) and synapses at the same time. Both interact in the same way in the overlay and exchange the same messages; moreover, those synapses can be members of several overlays independently (thus being able to replicate a request from one overlay to another) and can communicate with each other exclusively through a dedicated *Control Network*. The Control Network is basically a set of DHTs allowing each node to share routing information with other synapses without being aware of the routing of the undergoing message. So far the DHTs implemented are the following: (*i*) a Key table, responsible for storing unhashed keys circulating in the underlying overlays and every synapse accessing this table can easily retrieve the key in clear way using only the information it is aware of; (*ii*) a Replication table, in which is stored the number of times the key should be replicated across all of the the overlays; (*iii*) a Cache table, used to implement the replication of `GET` requests, and cache multiple responses and control the flooding of foreign networks.

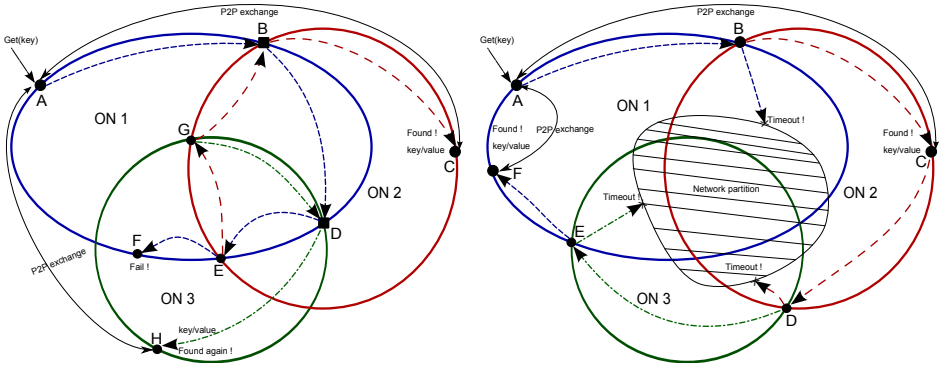


Figure 4.1: Routing across differents overlays and dealing with a network partition*

4.1.2 Routing across different intra-overlays

Figure 4.1 shows how a value present in one overlay can be retrieved from a `GET` launched by another overlay. Peer A in the overlay ON1 receives a `GET(key)` message: the routing goes until the synapse B, which triggers a second intra-overlay routing in ON2. The two routings proceed in parallel, and, in particular, the routing in ON2 terminates successfully with a peer-to-peer interaction between the peer A and peer C responsible of the resource. Routing continues on ON1 until synapse D, which triggers a third intra-overlay routing in ON3. The routing proceeds in parallel, and, in particular, routing in ON3 terminates successfully with a second peer-to-peer interaction between A and H, while routing in ON1 proceeds to a failure on peer F via the synapse E. Synapse E launches a fourth intra-overlay routing in ON2 that proceeds to a failure on node B (game over strategy) via synapse G. Finally, G launches a fifth intra-overlay routing on ON3, terminating with a failure on D (again game over strategy). Peers playing game over strategy are depicted as squares.

4.1.3 Dealing with network partition

Figure 4.1 also shows how intra-overlays take advantage of joining each other in order to recover situations where network partitioning occurs (because of the partial failure of nodes or the high churn of peers). Since network partitions affect routing performance and produce routing failures, the possibility of retrieving a value in a failed intra-overlay routing is higher, thanks to alternative inter-overlay paths. More precisely, the figure shows how a value stored in peer E of the overlay ON1 can be retrieved in presence of a generic network partition by routing via ON2 and ON3 through synapses B,C,D, and E.

4.2 Description of the Synapse Protocol Using the ASM Formalism

Section 3.1 describes the Chord protocol in the setting of ASM, and gives proof of the conditions under which a system maintained by this protocol forms stable

and correct structure and distributes the keys over the nodes. The Synapse protocol is not fully exhaustive and its underlying structure depends on the protocol of all overlay networks participating the system. Thus, the specification of the Synapse protocol in ASM will be presented here as an extension of the description in Chapter 3.1.

Let K, J, N and M be three positive integers.The following disjoint universes will be introduced:

- the set $Network = \{net_1, \ldots, net_N\}$ denotes all of the overlay networks present in the given system,
- the set $Hash = \{hash_1, \ldots, hash_N\}$ denotes hash functions for each of the overlay networks,
- the set $Node = \{node_1, \ldots, node_M\}$ represents the set of all of the nodes participating in the given system,
- the set $Key = \{key_1, \ldots, key_K\}$ denotes of identifiers of objects that might be stored in the considered system, and the set $Value = \{value_1, \ldots, value_K\}$ represents the values of those K objects,
- the set $Query = \{query_1, \ldots, query_J\}$ denotes all possible queries in the given system,
- the set $Action = \{join, leave, syn\_get, syn\_put\}$ denotes of the possible actions of a synapse node.

Each network in $Network$ is equipped with its own specific Join, Leave, Put and Get rules. As these rules depend on the protocols in each of the intra-overlays, they cannot be specified formally at this point.

Also, the following functions are introduced:

- $action : Node \rightarrow Action$, to save current action of a node,
- $networkList : Node \rightarrow ListOfNetworks$, which maps every $node \in Node$ into a list of overlays in which that $node$ participates,
- $processed : Node \rightarrow ListOfQueries$, to register already processed queries.
- $keyTable : Node \times Network \times Hash \rightarrow Key$, for connecting hashed and unhashed values of the keys for every overlay network,
- $cacheTable : Network \times Key \rightarrow ListOfValues$, used for caching already returned values.

The last two functions will be used only for the *black box* Synapse protocol.

4.2.1 Rules

During each execution of a *Synapse_ agent Module*, which is defined in Section 4.2.2 below, the rules ReadMessages, SynGet and SynPut will be applied. The responsibility of the ReadMessages rule is to process all of the messages sent to a particular node:

```
ReadMessages=
Read Messages Dedicated To Me,
Change Local Variables If It Is Requested And
Clear Processed Messages
```

If a synapse node applies one of the SynPut or SynGet rules, the main operation is to invoke the Put or Get rules, respectively, of the underlying protocols. If a synapse node is queried by some other node, the same procedure extends the search space and, possibly, returns more answers. The *white* and *black box* models of these rules slightly differ. Namely, in the case of the White Box model, the functions *keyTable* and *cacheTable* are not used because all of the nodes are aware of the changes made to the original protocol, while in the *black box* model the synapses need those functions in order to access and manipulate the unhashed keys. Here, only the high level version of these rules will be given. For the detailed version of the rules, all of the protocols that are used by all overlay networks has to be known and then, also, to change the rules of the basic protocols, i.e. to change all of the rules that are given in Appendix A.

White box Synapse Get and Put in ASM

```
SynPut=
forall net with net ∈ networkList(Me)
  Invoke Put Of Network net To Store ⟨key, value⟩
```

```
SynGet=
if query ∉ processed(Me) and ttl > 0          Check if the query is already
                                               processed or TTL is reached
  forall net with net ∈ networkList(Me)
    par
      Invoke Get Of Network net To Find key
        with Reduces ttl
      processed(Me).add(query)
    endpar
```

Black box Synapse Get and Put in ASM

```
SynPut=
seq
  Get Unhashed Value of key From keyTable
  forall net with net ∈ networkList(Me)
    Invoke Put Of Network net To Store ⟨key, value⟩
endseq
```

```
SynGet=
seq
  Invoke Get In Original Network
  Get Unhashed Value of key From keyTable
  if query ∉ processed(Me) and ttl > 0        Check if the query is already
                                               processed or TTL is reached
    par
```

```
      Get Results From The cacheTable          Check if a similar query
                                                got the result
      forall net with net ∈ networkList(Me)
          par
           Invoke Get Of Network net To Find key
             with Reduces ttl
           processed(Me).add(query)
         endpar
    endpar
  Add Result To cacheTable                     Store the result for future queries
endseq
```

4.2.2 Synapse module

The following main module contains actions executed by every synapse node.

```
seq
  ReadMessages                                 Process messages
  Choose An Action                             Choose next action
  par
    if action(Me) = join
      seq
        Choose net To Join
        par
          Join network net                     Invoke Join of net
          networkList(Me).add(net)             Add net to local list
        endpar
      endseq
    endif
    if action(Me) = leave then
      seq
        Choose net To Leave
        par
          Leave network net                    Invoke Leave of net
          networkList(Me).remove(net)          Remove net from local list
        endpar
      endseq
    endif
    if action(Me) = syn_put then
      SynPut                                   Invoke local SynPut
    endif
    if action(Me) = syn_get then
      SynGet                                   Invoke local SynGet
    endif
  endpar
endseq
```

This module is executed in an infinite loop, with the appropriate rule(s) being applied in each of the iterations.

4.3 Properties of the Synapse Protocol

In this section, the properties of the Synapse protocol will be examined. The focus will be on the probability assessments of the exhaustiveness of the Synapse protocol examined under various assumptions. Several software solutions will be used for the simulations and experiments:

1. openSynapse - *white box* model of the Synapse protocol based on [29],
2. jSynapse - both *white* and *black box* models developed by the members of the LogNet team, and
3. py-Synapse - Python scripts, using the *white box* model of the Synapse protocol.

The contribution of this thesis is given in Section 4.3.1. The other simulations and conclusions are the result of work by the other members of the LogNet team, and co-authors of [59], but they will presented here to give the "full" picture of the Synapse protocol.

4.3.1 Quassi-exhaustiveness of the Synapse Protocol

"Simple" probabilistic techniques will be used to solve this problems. An alternative approach could involve the techniques developed in random graph theory, which might seem as a natural path to take, given the complicated inter-connected structure of the network. Unfortunately, in this case the configuration is not fully random. On the contrary, certain parts exhibit a high level of structure. Another approach to avoid this situation and try to model our problem using random multigraphs and to focus on the existence of the paths of certain length, which is still an open problem.

First, Lemmas 4.3.1 and 4.3.2, respectively, will be used to give the probabilities of avoiding synapses in one overlay and avoiding all of the synapses which are members of another particular overlay. Then, the probability of exhaustiveness of the Pure *white box* model of the Synapse protocol will be given, where there is no failure of the nodes, TTL is not limited, and synapse nodes are members of exactly two different overlay networks. Afterwards, it will be examined how that probability changes if the failure of the nodes is allowed, the TTL is limited, and the higher degrees of connectivity of the synapses is allowed. Also, the probability of exhaustiveness of the Pure *black box* model of the Synapse protocol will be calculated.

To achieve statistical significance for the experiments and simulations performed in this Section, each configuration of the experiment or simulation was repeated between 1000 and 2000 times, depending on the experiment or simulation.Whenever where the settings of the experiment and simulation corresponds the setting of a theorem the results of obtained in [59] will be used.

4.3.2 Quassi-exhaustiveness of white box Synapse

Lemma 4.3.1. *Let there be b nodes in the overlay, where w of them are not synapses, while the rest of them are. If the search procedure were to contact up*

to l nodes (with uniform probability of choosing a number from $\{1,\ldots,l\}$), the probability of contacting no synapses is equal to:

$$P^{(1)}_{w,b,l} = \frac{1}{l}\sum_{m=1}^{l}\frac{\binom{w}{m}}{\binom{b}{m}}.$$

Proof. The probability of not contacting any of the synapses out of the m nodes that have been contacted is equal to:

$$P_m = \frac{\binom{w}{m}}{\binom{b}{m}},$$

because $\binom{n}{k}$ is the number of combinations in which k nodes out of the possible n could be contacted. From the set $\{1,\ldots,l\}$, it is possible to uniformly choose the number of nodes to be contacted, the final probability that is looked for is equal to:

$$P^{(1)}_{w,b,l} = \frac{1}{l}\sum_{m=1}^{l}P_m = \frac{1}{l}\sum_{m=1}^{l}\frac{\binom{w}{m}}{\binom{b}{m}}.$$

□

Lemma 4.3.2. *Let the system contain a number of overlays, and let M_0 and M_1 be two of these overlays. Let M_0 contain $b = w + r + g$ nodes, where w, r and g are the number of nodes that are not synapses, the number of synapses towards the overlay M_1, and the number of synapses to the remaining overlays in the system, respectively. If the search procedure were to contact up to l nodes (with uniform probability of choosing the number from $\{1,\ldots,l\}$), the probability of contacting no synapses to M_1, if it is known that at least one synapse has been contacted, is:*

$$P^{(2)}_{w,r,g,l} = \frac{1}{l}\sum_{m=1}^{l}\frac{\sum_{i=1}^{m}\binom{g}{i}\binom{w}{m-i}}{\sum_{i=1}^{m}\binom{g+r}{i}\binom{w}{m-i}}.$$

Proof. If m nodes have been contacted, the probability of not contacting any of the synapses leading to M_1, if at least one synapse has been contacted, is:

$$P_m = \frac{\sum_{i=1}^{m}\binom{g}{i}\binom{w}{m-i}}{\sum_{i=1}^{m}\binom{g+r}{i}\binom{w}{m-i}}.$$

Similarly to Lemma 4.3.1, from the set $\{1,\ldots,l\}$, it is possible to uniformly choose the number of nodes to be contacted, the final probability that is looked for is equal to:

$$P^{(2)}_{w,r,g,l} = \frac{1}{l}\sum_{m=1}^{l}P_m = \frac{1}{l}\sum_{m=1}^{l}\frac{\sum_{i=1}^{m}\binom{g}{i}\binom{w}{m-i}}{\sum_{i=1}^{m}\binom{g+r}{i}\binom{w}{m-i}}.$$

□

Hereinafter, it will be assumed that the complexity of the search procedure for any regular/control overlay is $log_2(n)$. Also the following notation will be used:

- $N, (N \geqslant 2)$ - total number of intra-overlays,
- n - number of nodes per intra- overlay,
- s - for the *white box* model - percentage of regular nodes that have become synapses; for the *black box* model - percentage of synapses with respect to all of the nodes in the system,
- p_f - probability for a node/synapse to fail,
- c - number of connections per synapse,
- F - the event that the given key has been found,
- KO - the event that the given key is stored at the same intra-overlay as the starting node,
- S - the event that a synapse has been contacted,
- D - the event that a query passed a maximum of D intra-overlays,
- B^c - the complementary event of some event B.

Pure white box Synapse

In this scenario, the nodes do not fail, TTL is not limited, and synapse nodes are members of exactly two different intra-overlay networks.

Theorem 4.3.1 (Pure white box Satisfaction Ratio)**.** *The probability for a node to get a value for a given key stored in the system, where all the synapses are connecting exactly two intra-overlays, is:*

$$P(F) = 1 - \frac{N-1}{N}\left(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{N-1},2ns\frac{N-2}{N-1},l}\right)^{N-1}\right),$$

where $l = \lfloor log_2((1+s)n) \rfloor$.

Proof. The probability $P(F)$ to find the given key is equal to:

$$P(F) = 1 - P(F^c).$$

There are two possibilities: the given key and the starting node are or are not stored in the same intra-overlay. Therefore, using the formula of total probability, it is possible to obtain:

$$P(F^c) = P(KO)P(F^c|KO) + P(KO^c)P(F^c|KO^c).$$

Due to the property of intra-overlays that if a key is stored in a particular intra-overlay it will always be found, $P(F^c|KO)$ is equal to 0, so:

$$P(F^c) = P(KO^c)P(F^c|KO^c).$$

Next, the probability that the given key and the starting node are not in the same intra-overlay is equal to:

$$P(KO^c) = \frac{N-1}{N}.$$

There are two cases in which the key that is stored in a different intra-overlay cannot be retrieved from the starting node. In the first case, none of the synapses have been reached during the search procedure at the starting intra-overlay. In the second case, at least one of the synapses was asked but anyway the key was not found (because the query didn't reach the overlay where the key is stored). This is reflected by:

$$P(F^c|KO^c) = P(S^c)P((F^c|KO^c)|S^c) + P(S)P((F^c|KO^c)|S),$$

where $P((F^c|KO^c)|S^c)$ is equal to 1. Finally:

$$P(F) = 1 - \frac{N-1}{N}(P(S^c) + P(S)P((F^c|KO^c)|S)). \tag{4.1}$$

From Lemma 4.3.1, it is possible to obtain $P(S^c)$. It is possible to consider the situation where the intra-overlay contains $(1+s)n$ nodes while $2sn$ of them are synapses, which lead to:

$$P(S^c) = P^{(1)}_{(1-s)n,(1+s)n,l}.$$

Also:

$$P(S) = 1 - P(S^c).$$

Similarly, from Lemma 4.3.2 it is possible to get $P((F^c|KO^c)|S))$. This time the situation where there is $N-1$ intra-overlays with $(1+s)n$ nodes in total, while $(1-s)n$ are not synapses, and $\frac{2ns}{N-1}$ are those synapses which lead to the particular overlay, is considered. This has to be adopted for all $N-1$ overlays that are not in the starting network, yielding:

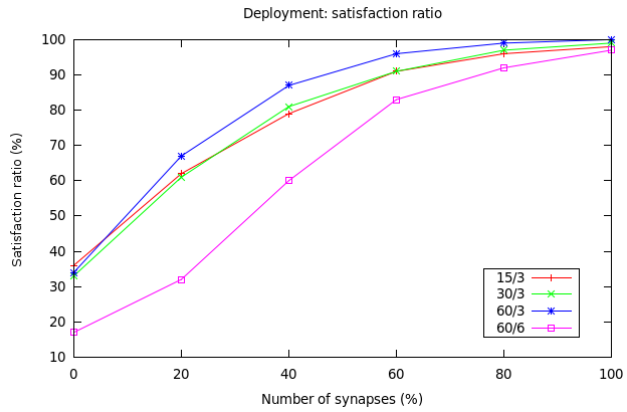
$$P((F^c|KO^c)|S)) = P^{(2)}_{(1-s)n,\frac{2ns}{N-1},2ns\frac{N-2}{N-1},l}{}^{N-1}.$$

Now, all of the components required for equation (4.1) are obtained, so:

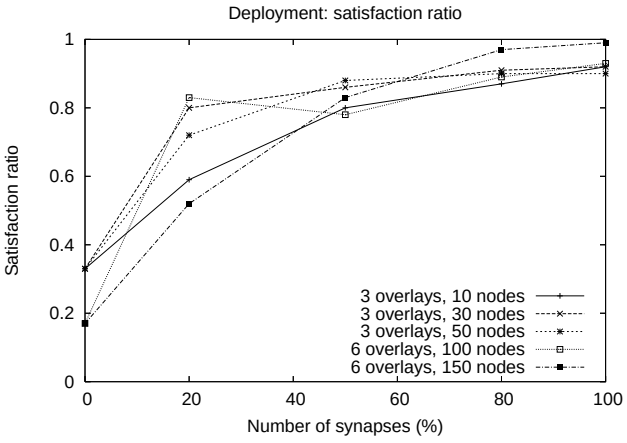
$$P(F) = 1 - \frac{N-1}{N}\left(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{N-1},2ns\frac{N-2}{N-1},l}\right)^{N-1}\right).$$

□

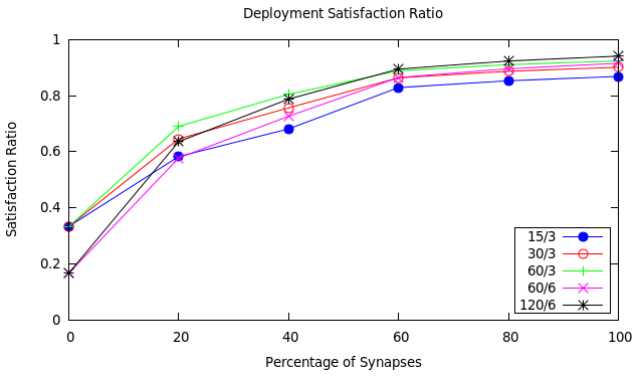
In Figure 4.2, the results of the deployment of openSynapse and JSynapse are presented, as well as the graph constructed from the result of Theorem 4.3.1. The lines represent various experiments where the given number of nodes was uniformly distributed over the given number of overlay networks, as described in the corresponding legends. The percentage of the nodes that have become synapses is given on the x-axis. It is possible to see from the graphs that there exists a substantial correspondence between the theoretically predicted results and those obtained through experimentation.



(a) openSynapse Tests



(b) JSynapse Tests*



(c) Theoretical Calculations

Figure 4.2: Satisfaction Ratio - Experiments and Theory

White box Synapse with Node Failure

In this scenario, with respect to the previous one, it will be allowed the possibility for a node to fail.

Lemma 4.3.3. *The expected number of nodes per intra-overlay is:*

$$\mathbb{E}(n) = (1 - p_f)n.$$

If node failures are allowed in Theorem 4.3.1, the number of nodes per intra-overlay would be $\mathbb{E}(n)$ rather then n. Also, the probability that a given key was not stored on some of the nodes that have failed is $1 - p_f$. With this, the following result is obtained:

Theorem 4.3.2. *The probability for a node to get a value for a given key stored in the system is:*

$$P(F) = (1 - p_f)\Bigg(1 - \frac{N-1}{N}\Bigg(P^{(1)}_{(1-s)(1-p_f)n,(1+s)(1-p_f)n,l} + \\ + \Big(1 - P^{(1)}_{(1-s)(1-p_f)n,(1+s)(1-p_f)n,l}\Big)\left(P^{(2)}_{(1-s)n,\frac{2(1-p_f)ns}{N-1},2(1-p_f)ns\frac{N-2}{N-1},l}\right)^{N-1}\Bigg)\Bigg),$$

where $l = \lfloor log_2((1+s)n) \rfloor$.

White box Synapse with Multiple Connectivity of Synapses

In this scenario, it is allowed synapses to connect more than two intra-overlays at the same time, but, it won't be consider node failures.

Theorem 4.3.3. *The probability for a node to get a value for a given key stored in the system, where all the synapses are connecting exactly c intra-overlays, is:*

$$P(F) = 1 - \frac{N-1}{N}\Bigg(P^{(1)}_{(1-s)n,(1+(c-1)s)n,l} + \\ + \Big(1 - P^{(1)}_{(1-s)n,(1+(c-1)s)n,l}\Big)P^{(2)}_{(1-s)n,\frac{cns}{N-1},cns\frac{N-2}{N-1},l}{}^{N-1}\Bigg),$$

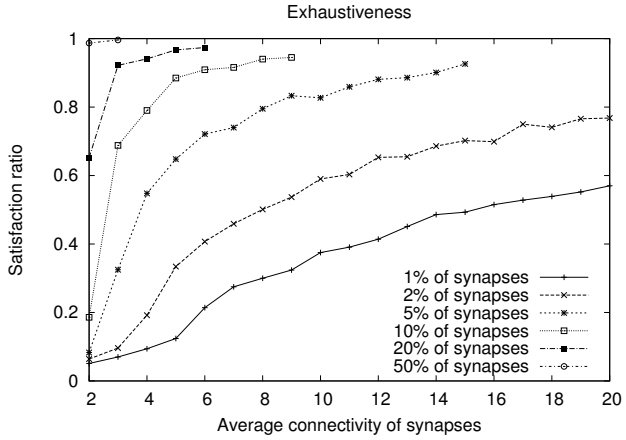
where $l = \lfloor log_2((1+(c-1)s)n) \rfloor$.

Proof. The total number of nodes in Lemmas 4.3.1 and 4.3.2 increases with every new connection of a new synapse, so we can consider every new connection as a new node of the intra-overlay network. □

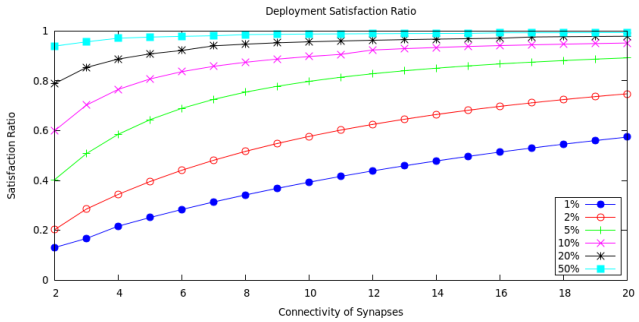
In Figure 4.3, the situation where 10000 nodes are uniformly distributed over 20 intra-overlay networks is illustrated. The lines represent the percentage of the nodes which are transformed to synapses, while on the x-axis the degree of connectivity of the synapses is presented. Again, it is possible to see that theoretical predictions correspond to the obtained experimental results.

White box with TTL

In this scenario, it will be considered TTL stopping criterion for the issued request. Under assumption that TTL is the number of overlays that can be reached during one query, then the following theorem holds:



(a) py-synapse Simulations*



(b) Theoretical Calculations

Figure 4.3: Connectivity of Synapses - Simulations and Theory

Theorem 4.3.4. *The probability for a node to get a value for a given key stored in the system is:*

$$P(F) = 1 - \frac{N-1}{N}\left(\frac{D}{N}\left(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{D-1},2ns\frac{D-2}{D-1},l}\right)^{D-1}\right)\right) + \frac{N-D}{N}\right)$$

where $l = \lfloor log_2((1+s)n) \rfloor$, and D is the maximum allowed number of intra-overlays for a query to pass.

Proof. Similarly to the proof of Theorem 4.3.1, it is possible to get:

$$P(F) = 1 - \frac{N-1}{N} P(F^c|KO^c),$$

where:

$$P(F^c|KO^c) = P(D)P((F^c|KO^c)|D) + P(D^c)P((F^c|KO^c)|D^c).$$

Next, since the probability $P((F^c|KO^c)|D^c)$ of not finding the given key is 1, and $P(D) = \frac{D}{N}$, the following holds:

$$P(F^c|KO^c) = \frac{D}{N}P((F^c|KO^c)|D) + \frac{N-D}{N}.$$

Now, like in the proof of Theorem 4.3.1, the event $(F^c|KO^c)|D$ can be divided into whether the synapse in the starting network has or has not been contacted, but this time taking into account a system containing D overlays:

$$P(F^c|KO^c)|D) = P^{(1)}_{(1-s)n,(1+s)n,l} + (1 - P^{(1)}_{(1-s)n,(1+s)n,l})P((F^c|KO^c)|D)|S^c),$$

and

$$P((F^c|KO^c)|D)|S^c) = P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{D-1},2ns\frac{D-2}{D-1},l}\right)^{D-1},$$

which completes our equation. □

In Figure 4.4, a system of 1000 nodes uniformly distributed over 10 intra-overlay networks is examined. The lines on the graphs represent scenarios with different percentages of nodes that have become synapses, whereas the x-axis is dedicated to the TTL, again with a clear correspondence between the theory and the experiments.

4.3.3 Quassi-exhaustiveness of black box Synapse

Theorem 4.3.5 (Pure black box Satisfaction Ration)**.** *The probability for a node to get a value for a given key stored in the system, in the Pure black box Synapse model, is:*

$$P(F) = 1 - \frac{N-1}{N}\left(P^{(1)}_{(1-s)n,n,l} + \left(1 - P^{(1)}_{(1-s)n,n,l}\right)\right) \cdot \cdot P^{(2)}_{(1-s)n,\frac{sn}{N-1},sn\frac{N-2}{N-1},l}{}^{N-1} P^{(1)}_{sn(N-1),snN,L}.$$

where $l = \lfloor log_2(n) \rfloor$ *and* $L = \lfloor log_2(snN) \rfloor$,

Proof. Similarly as in the proof of Theorem 4.3.1, it is possible to obtain:

$$P(F) = 1 - \frac{N-1}{N}(P(S^c) + P(S)P((F^c|KO^c)|S)). \tag{4.2}$$

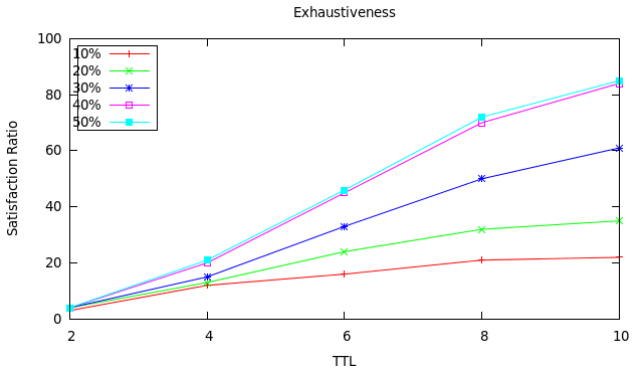
Also, the equations

$$P(S^c) = P^{(1)}_{(1-s)n,n,l},$$

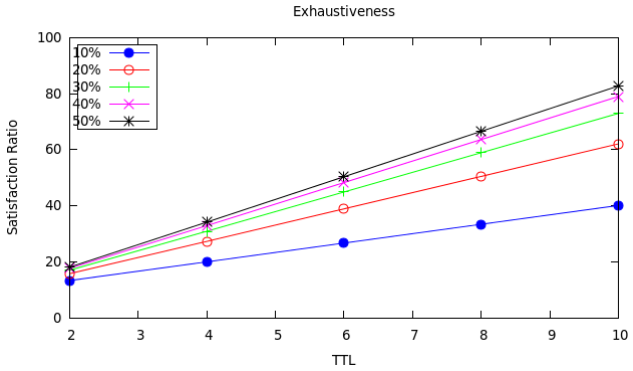
and $P(S) = 1 - P(S^c)$ hold. This time, besides the N intra-networks, there is one more control network consisting of all of the synapses in the system. Therefore, the following:

$$P((F^c|KO^c)|S)) = \left(P^{(2)}_{(1-s)n,\frac{sn}{N-1},sn\frac{N-2}{N-1},l}\right)^{N-1} P^{(1)}_{sn(N-1),snN,L}$$

completes Equation 4.2. □



(a) py-synapse Simulations



(b) Theoretical Calculations

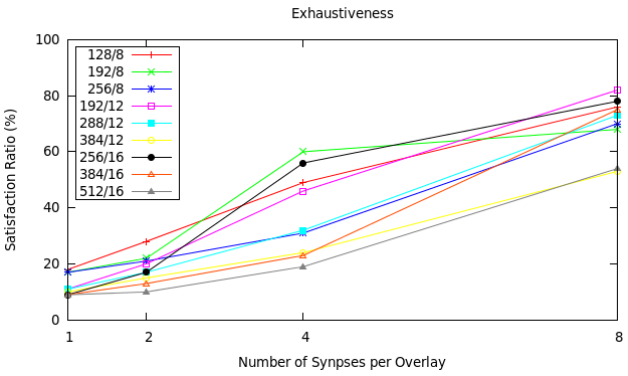
Figure 4.4: TTL - Simulations and Theory

Similarly, like in the previous section, it is possible to extend Pure *black box* Synapse model and the obtained formula by allowing node failures and multiple connectivity of the synapses, or by introducing TTL. In Figure 4.5, the results of the deployment of jSynapse are shown, as well as the graph illustrating the result of Theorem 4.3.5. The lines represent various experiments where the given number of nodes was uniformly distributed over the given number of intra-overlay networks, while the total number of synapses per overlay is given on the x-axis. Just as in the previous three cases, it is possible to notice a clear correspondence between the graphs.

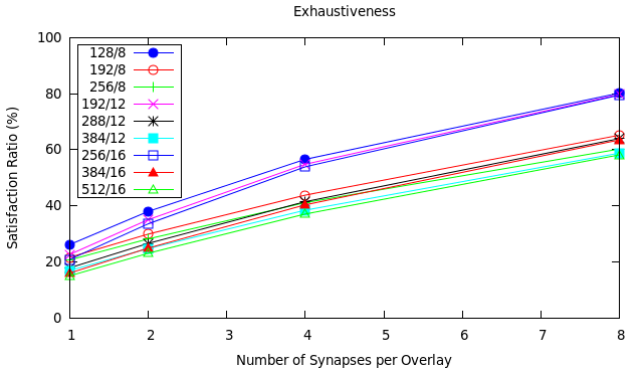
4.3.4 Latency and Communication

To be able for better understanding of the behavior of platforms interconnecting structured overlay networks through the Synapse approach the series of simulation were realized.

The focus was on the key metrics traditionally considered in distributed information retrieval process, such as latency (number of hops required to reach



(a) jSynapse simulations



(b) Theoretical Calculations

Figure 4.5: Black box Synapse - Simulations and Theory

the requested object) and the amount of communications produced (number of messages generated for one request). The behavior of these metrics will be highlighted while varying the topology (the number of synapses and their connectivity, TTL, the number of intra-overlays ...).

Settings

The topology of the intra-overlay simulated is a set of Chord networks interconnected by some synapses. Information is a set of $\langle key, value \rangle$ pairs. Each pair is unique and exists once and only once in the network. The unstructured interconnection of structured networks will be studied with discrete-time simulation: queries are launched on the first discrete time step, initiating a set of messages in the network, and each message sent at the current step will be received by its destination (next routing hop) at the next hop.

Latency

The first set of simulations has the intent of studying how the previously mentioned metrics vary while increasing the number of synapses or the degree of existing ones (the number of intra-overlays a co-located node belongs to). The

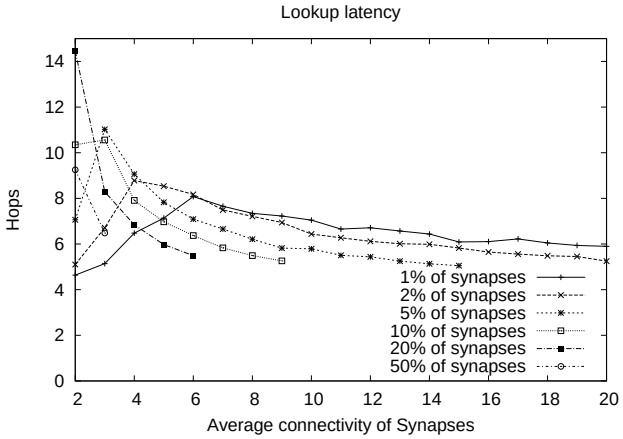


Figure 4.6: Latency in Synapse*

number of nodes is fixed to 10000, uniformly distributed amongst 20 intra-overlays (approximately 500 nodes within each Chord). Queries are always triggered by one random node, the key sought by a query is also picked uniformly at random among the set of keys stored by the network. A query is said to be *satisfied* if the pair corresponding to the key has been successfully retrieved.

As illustrated in Figure 4.6, one first point to notice is that the number of hops remains logarithmic when changing a Chord network into a Synapse network (the number of nodes is 10000, the latency never exceeds 14). Other experiments conducted by increasing the number of nodes confirm this. More precisely, Figure 4.6 highlights the following behavior: (*i*) when the network contains only a few synapses, the latency first increases with the degree of synapses: only a few *close* keys are retrieved (keys available in the network of the node that initiated the query); (*ii*) then, when both parameters (the connectivity and the number of synapses) have reached a certain threshold, the searches can touch more synapses, and the whole network becomes progressively visible, multiple parallel searches become more and more frequent and distant nodes (and keys) are reached faster. As it is possible to see, increasing the number of synapses decreases the latency of only a small constant factor. In other words, synapse topologies do not need a lot of synapses to be efficient. This result fits with random graphs behavior: when the number of neighbors in the graph reaches a (small) threshold, the probability for the graph to be connected tends towards 1.

Communication

Obviously, multiple searches in parallel lead to an increased number of messages. As illustrated in Figure 4.7, this number increases proportionally with the connectivity and the number of synapses.

The number of messages can become high when the number of synapses increases. To limit this impact, TTL is introduced to reduce the overhead while keeping an acceptable level of exhaustiveness. A second set of experiments examines the impact of the TTL on the search queries. This TTL is simply

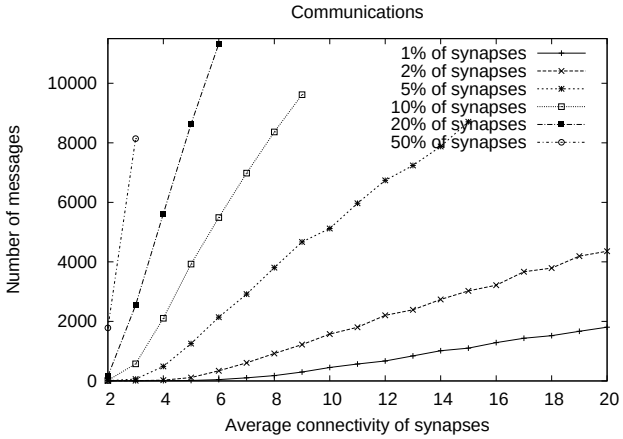


Figure 4.7: Communications in Synapse*

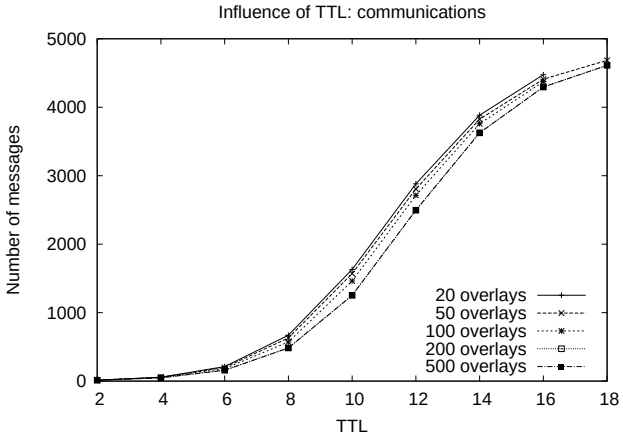


Figure 4.8: TTL and communications*

decreased every time the query traverses a node.

The number of overlays is varying, to experiment the impact of the *granularity* of the network. In other words, a Synapse network made of few large structured intra-overlays could be called *strongly structured*, while another network with many smaller structured intra-overlays could be called *weakly structured*. The number of nodes was still set to 10000, and every node is a synapse belonging to 2 intra-overlays chosen uniformly at random.

As highlighted by Figure 4.8, it is possible to drastically reduce the amount of communications experienced, with the number of messages being almost divided by 2. To sum up, Synapse architectures can use TTL, leading to a significant exhaustiveness while drastically reducing the expected overhead. Finally, still see Figure 4.8, the *granularity* does not significantly influence communications when the number and connectivity of the synapses are fixed.

4.4 Related Work

Several propositions have been made over the years to build alternate topologies based on the coexistence of smaller local overlay networks. The first approach has been based on hierarchical systems [37, 84], where some elected super-peers being are promoted to a top-level overlay network, leading to the requirement of costly merging mechanisms to ensure a high level of exhaustiveness. In a more general view, merging several co-existing structured overlay networks has been shown to be a very costly operation [25, 74].

In the context of mobile ad hoc networks, Ariwheels [13,57] has been designed to provide a large variety of services through a multi-layer overlay network, where super-peers, called Brokers, act as servers for a subset of peers. Ordinary peers, called Agents, submit queries to their Broker and receive results from it. Ariwheels provides an efficient mapping between physical devices in the wireless underlay network and virtual entities in the overlay network.

Authors in [23] present two models for two overlays to be (de)composed, known as *absorption* (a sort of merging) and *gatewaying*. Their protocol enables a CAN-network to be completely absorbed into another one (in the case of the absorption), and also to provide a mechanism to create bridges between DHTs (in the case of the gatewaying). They do not specifically take advantage of a simple assumption that nodes can be part of multiple overlays at the same time, thus playing the role of natural bridges.

More recently, authors in [22] propose a novel information retrieval protocol, based on gateways, called *DHT-gatewaying*, which is scalable and efficient across homogeneous, heterogeneous and assorted co-existing structured overlay networks[1]. They argue that there is not one preferred structured overlay network implementation, and that peers are members of co-existing DHTs. Their assumptions are (*i*) only some peers support the implementations of different DHTs and (*ii*) some peers are directly connected to peers that are members of other DHTs, and are called *Virtual Gateways (VG))*. When a request is sent in one overlay, and no result was found, the requester can opt to widen his search by forwarding the original search request to nodes which belong to other structured overlay networks (mapping the search to the format supported by their relative overlay). A TTL value is added to the original search in order to avoid cycles; this value is decremented each time a request crosses a new DHT domain. Unfortunately the evaluation of their protocol lacks precious details and precision. It is unclear how they evaluate their protocol.

Authors in [53] present Synergy, an overlay inter-networking architecture which improves routing performance in terms of delay, throughput and packet loss by providing cooperative forwarding of flows between networks. Authors suggest that co-located nodes are good candidates for enabling inter-overlay routing and that they reduce traffic. Our approach can also be seen as a deeper study of their concepts.

On the way of designing inter-overlay networking based on co-located nodes, authors in [50] present algorithms which enable a symbiosis between different overlays networks with a specific application in mind: file sharing. They propose mechanisms for hybrid P2P networks cooperation and investigate the influence

[1] Ex. Two 160-bit Chord, or two 160/256-bit Chord, or one 160-bit Chord and one 256-CAN.

of system conditions, such as the numbers of peers and the number of meta-information a peer has to keep. They provide interesting observations on how to join a candidate network, cooperative peers' selection, how to find other P2P networks, when to start cooperation, by taking into account the size of the network (for instance, a very large network will not really benefit from a cooperation with a small network), so on and so forth. Again, a more comprehensive understanding of this approach is missing.

Authors in [42] consider multiple spaces with some degree of intersection between spaces, i.e. with co-located nodes. They focus on different potential strategies to find a path to another overlay from a given overlay, i.e. how requests can be efficiently routed from one given overlay to another one. They compare various inter-space routing policies by analyzing which trade-offs, in terms of state overhead, would give the best results, in terms of the number of messages generated and routed, the number of hops it takes to find a result and the state overhead (i.e. the number of fingers a node has to keep). They provide a comparative analytical study of the different policies. They show that with some dynamic finger caching and with multiple gateways (in order to avoid bottlenecks and single points of failures) which are tactfully laid out, they obtain pretty good performances. Their protocol focuses on the interconnection of DHTs, while Synapse can be extended it to any kind of overlays.

In preliminary work [58] of INRIA LogNet team, BabelChord protocol was introduced as solution for inter-connecting Chord overlay networks using co-located nodes that are part of multiple Chord "floors". These nodes connect, in an unstructured fashion, several Chord overlays together The simulations showed that it is possible to achieve pretty high exhaustivity with a small amount of those co-located nodes.

In [2], the authors have developed a multi-ring model based on Chord, in which each shared resource is described by one or more keywords. Nodes are organized in multiple keyword rings, and each node in a keyword ring contains the list of nodes that host resources matching a certain keyword/value pair. A new keyword ring is only created when the number of queries or registered resources for the keyword rises above a certain threshold. To enable keyword rings to be found, a Super Ring is used to host a ring of nodes which contain pointers to other rings. One major drawback of the model is that it heavily depends on the bootstrap node.

In [60], the developers present ML-Chord, a multi-layered P2P resource sharing model. They introduce overlay layers of categories. The number of these categories depends on the number of categories for a specific domain or ontology. Also, they introduce two types of nodes: *normal peers*, which can be associated with one or several layers and *bridge peers*, which are peers with better capabilities which are linked to all categories, and which themselves form a category as well. The problem with this approach is that it is not possible to simply encapsulate a new system into an existing one because all of the Chord layers share the same hash function. Although this system is scalable and efficient, it is not possible easy to introduce a new category during the system lifetime. The developers suggest that one node should be linked to only one layer for better performance. So, if a node with good capabilities has not become a bridge peer at the start of the lifecycle of the system, it will remain a normal node, and its beneficial capabilities will be lost.

5

Distributed Catalog of Serbian Digitized Cultural Collections

This chapter presents recommendation for two metadata formats for describing the digitized objects and collections of the cultural and scientific heritage in Serbia. The full description of these recommendations is available at `http://www.ncd.org.rs/ncd_en/standards.html`. Based on these recommendations and the Chord protocol the Distributed Catalog of Serbian Digitized Cultural Collections has been developed. A description of this application will be given here. The realized application can be easily adopted to use Synapse protocol, but at this stage, when few collections have been described and almost all of them are from the librarian world, the decision was to use Chord protocol. The catalog is active at `http://digitalne_kolekcije.ncd.org.rs/web_search/Search_jsp.jsp`.

5.1 Recommendation for the National Standard for Describing Digitized Heritage in Serbia

For the recommendation for the National Standard for Describing Digitized Heritage in Serbia some requirements were determined to accommodate more-or-less divergent descriptive practices and to articulate needs of different specific heritage contexts. More precisely, the requirements were:

- the definition should be rich enough so that it could be used to describe assets from libraries, museums, and archives, as well as from the other providers (research institutions, for example),
- the definition should be flexible enough to allow translation to and from international standards,
- the definition should allow: multilingual description of cultural and scientific assets and use of some predefined dictionaries for the particular elements as much as possible.

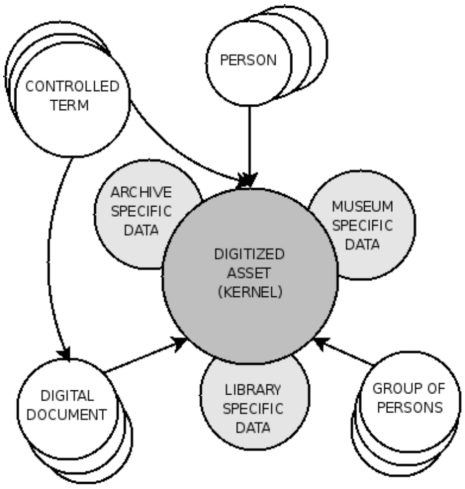


Figure 5.1: Objects of NCD Recommendation for Description Digitized Cultural Heritage

The underlying idea is that the description of cultural assets based on this metadata set should be used to make decisions about future digitization. For this reason, the proposal contains more structured and detailed elements than it is usually the case.

5.1.1 Description of the standard

According to the previous requirements, the following five basic objects were distinguished:

- *Person*, and *Group-of-persons* for entities that have intellectual or some other important contribution to the creation of assets, that are subjects of and/or owners of cultural heritage assets,
- *Digital-document*, for an individual file containing digital representation of the corresponding asset,
- *Digitized-asset*, for the digitized asset of heritage, and
- *Controlled-term*, for elements of a predefined dictionary.

Figure 5.1 contains a structure based on the above mentioned set of metadata objects, where the nodes represent the objects, and the arcs represent relationships between them. This structure forms a unique core representing an intersection of descriptions of assets from libraries, museums, and archives. However, our proposal allows some extensions related to the particular fields of interests.

The object Controlled-term corresponds to standardized data values that can be used to improve access to information about heritage. Those values should be organized in a structured controlled vocabulary which offers preferred terms and synonyms, as for example in [3].

The sets of meta-data that corresponds to each of these objects were defined. They contain a descriptive and an administrative part. The administrative part identifies authors and/or owners of descriptions, assets of heritage and their digital representations. Note that, if appropriate, a description of access rights is also available to limit access to (parts of) resources. The following provides an abbreviated version of the descriptions of the mentioned objects[1] (the full description in Serbian is available [69]).

The object Person

Metadata which describe the object Person are:

- Descriptive:
  - Name
    - First name
    - Middle name
    - Family name
  - Version of name ...
    - First name
    - Middle name
    - Family name
  - Nickname ...
  - Pseudonym ...
  - Day of birth
  - Day of death
  - Sex
    - Type
    - Period
  - Biography ...
    - Biography
    - Language of biography
  - Resources related to the person ...
  - Picture(s) ...
  - Note ...
    - Note
    - Language of note
- Administrative:
  - Record creation date
  - Record creator(s)
  - Record owner

[1] '...' means that the element could be repeated more than once.

The object Group-of-persons

Metadata which describe the object Group-of-persons are:

- Descriptive:
  - Name
  - Version of name ...
  - Foundation date
  - Dismissing day
  - History
  - Place
  - Activity ...
  - Type
  - Identifier ...
  - Description ...
    - Description
    - Language of description
  - Resources related to the group of persons ...
  - Picture(s) ...
  - Member ...
    - Identifier
    - Role
  - Note ...
    - Note
    - Language of note
- Administrative:
  - Record creation date
  - Record creator(s)
  - Record owner

The element Activity specifies in more details fields in which the group is active, while the elements Type and Identifier identify the group according to the local codification system.

The object Digital-document

Metadata which describe the object Digital-document are:

- Descriptive:
  - Title ...
    - Title
    - Language of title

    - Creator ...
        - Identifier
        - Role
    - Location of digital document ...
    - Related asset ...
    - Note ...
        - Note
        - Language of note

- Administrative:
    - Archive date
    - Digital document format(s) ...
    - Size
    - Digital document MIME format(s) ...
    - Capture device(s) ...
    - Rights
    - Access rights
    - URL ...
    - Digital object owner
    - Record creation date
    - Record creator(s)
    - Record owner

The element Related asset of heritage connects a particular digital document to the object(s) from which it is produced (note that one digital document can represent more than one physical object - for example, it is possible that two or more objects is represented by one photo). The administrative part for the object Digital-document contains more elements than it is the case for the other objects: they describe history, technical details and operations performed during the digitization process itself.

The object Digitized-asset

Metadata which describe the object Digitized-asset are:

- Descriptive:
    - Title
        - Title
        - Original name
        - Version of name ...
            - Version of name
            - Type of version
            - Language of version

- Creator ...
  - Identifier
  - Role
- Contributor ...
  - Identifier
  - Role
- Subject ...
  - Topic
  - Spatial
  - Temporal
- Classification ...
  - Classification scheme
  - Classification identifier
  - Classification group
- Description ...
  - Description
  - Language of description
- Resources related to the object ...
- Category
  - Date of proclamation
  - Decision identifier for the proclamation
  - The proclamation decision made by
  - National category
  - Date of national categorization
  - Decision identifier for the national categorization
  - The national categorization made by
  - International category ...
    - Category
    - Date
    - Decision identifier for the international categorization
    - The international categorization made by
- Provenance
  - Date of origin
  - Provenance of origin ...
  - Version ...
    - Description of the version ...
      - Description
      - Language of description
    - Date of the version
    - Place of the version
    - Country of the version

    - Physical description ...
        - Dimension type
        - Dimension value
        - Description
    - Material ...
    - Type
    - Acquisition
        - Date
        - Type
    - Related object ...
        - Type of relation
        - Related object ID
    - History ...
        - Object's history
        - Language of history
    - Owner
    - Source object ID
    - Bibliography ...
    - Note ...
        - Note
        - Language of note
- Administrative:
    - Rights
    - Access rights
    - Record creation date
    - Record creator(s)
    - Record owner

The elements Broader object and Related object describe relationships between parts of compound objects.

The object Controlled-term

Metadata which describe the object Controlled-term are:

- Descriptive:
    - Accepted form of the term
    - Explanation of the accepted form of the term
    - Synonym
    - Translation to other schemata/languages
    - Description in Serbian

    - Broader term
    - Related term
    - Note ...
        * Note
        * Language of note

- Administrative:
    - Record creation date
    - Record creator(s)
    - Record owner

The proposed structure of this object could be useful in the process of creating a structured controlled thesaurus of terms, synonyms, classification schemes, etc. The element Translation to other schemata/languages should allow use of recognized standard multilingual terminological sources.

Similarly as above, the elements Broader term and Related term correspond to relationships between terms.

5.1.2 Translations to international standards

As a particular part, the proposal defines mappings to and from some international standards for meta-data (Dublin Core Metadata Element Set, The European Library Application Profile, Table of Core Metadata Elements for Library of Congress Digital Repository Development, Encoded Archival Description). An example of the translation of the elements of the objects Digital-document and Digitized-asset is given in Table 5.1. As it can be seen, some different elements of our objects must be grouped into one element and for some elements there are no corresponding translations in some of the mentioned standards. The full mapping is available at `http://www.ncd.org.rs/ncd_sr/standards/ncd_dokumenti_2_0.html#_Toc160349341`.

5.1.3 Library Specific

Metadata which describe the object Digitized-asset specific to the assets from the libraries are:

- Language ...
    - Language
    - Type
- Publication
- Edition
- Identifier ...
- Table of content
- Collection ...

NCD	DC	ISAD	EAD	TEL AP	Library of Congress
Description	description	3.3.1	scopecontent	description	description summary
Type	type	3.1.5	genreform	type	original content type
Material	medium	3.1.5	physdesc	format.medium	
Rights	rights	3.4.2	userestrict	rigths	access rights
Access rights	accessRights	3.4.1	accessrestrict	rights	access category
Note		3.6.1	note, odd		
Capture device(s)			daodesc		capture device ID
MIME format(s)	Format		daodesc	format	internet media type

Table 5.1: Partial Mappings to/from International Standards from Section 2.3.2

- Printer
- Place of printing
- Publisher ...
- Place of publishing
- Issued
- COBISS identifier

5.1.4 Archival Specific

Metadata which describe the object Digitized-asset specific to the assets from the archives are:

- Level
- Acquisition
- Appraisal
- Accruals
- Physical and technical description
- Origin location
- Alternative forms
- Rules of keeping

5.1.5 Museum Specific

Metadata which describe the object Digitized-asset specific to the assets from the museums are:

- Number of parts
- Epoch
- Author's signature
- Monument Text
- Technics ...
- Style ...
- Finding ...
    - Part ...
    - Date ...
    - Place ...
        - Name
        - Part
        - Coordinates
        - Settlement
        - Municipality
        - Region
        - Country

Example 5.1.1. *In this example it is illustrated how to describe a book of a well-known Serbian comediograph Branislav Nušić. The book, Autobiography, belongs to the collection of the Serbian Children's digital library [73]. The illustrator of the edition, published by Kreativni centar, is Dobrosav Živković, and the editors are Simeon Marinković and Slavica Marković. Digitization of the book was performed by Nikola Pavlović from National library of Serbia. The author of the metadata records is Tamara Butigan Vučaj.*

Table 5.2 contains a record which corresponds to the author. Table 5.3 contains a record which corresponds to the digital version of the book. Table 5.4 contains a record which corresponds to the "hardcopy" version of the book.

Descriptive	
Name	
* First name	Branislav
* Family name	Nušić
Name version	
* First name	Ben
* Family name	Akiba
Date of birth	1864
Date of death	1938

Sex	male
Biography	
* Link * Language of Biography	http://sr.wikipedia.org/wiki/Branislav_Nušić sr
Biography	
* Link * Language of Biography	http://en.wikipedia.org/wiki/Branislav_Nusic en
Administrative	
Record Creation Date	30/3/2007
Record Creator	Tamara Butigan Vučaj
Record Owner	NBS

Table 5.2: Description of the object Person

Descriptive	
Title	Digitalizovana dečija knjiga Autobiografija, Branislava Nušića
* Version of Title * Language of Version	Digitized children's book: Autobiography written by Branislav Nušić en
Creator	
* Identifier	ID (Nikola Pavlović)
* Role	technician
Location of digital document	http://www.digitalna.nb.rs/wb/NBS/Knjige/Srpska_decja_digitalna_biblioteka/II-449580
Administrative	
Archive date	2004
Digital document format	jpg
Size	17.9MB
Digital document MIME format	image
Capture device	Epson GT 15000
Rights	NBS gained rights from the publishing house
Access rights	Unlimited
Digital object owner	NBS
Record creation date	30/3/2007
Record creator	Tamara Butigan Vučaj
Record owner	NBS

Table 5.3: Description of the object Digital-document

Descriptive	
Title	
* Title	Autobiografija

* Original name	Autobiografija
* Version	Autobiography
* Language of Version	en
Creator	
* Identifier	ID (Branislav Nušić)
* Role	writer
Contributor	
* Identifier	ID (Dobrosav Živković)
* Role	illustrator
Contributor	
* Identifier	ID (Simeon Marinković)
* Role	editor
Contributor	
* Identifier	ID (Slavica Marković)
* Role	editor
Classification	
* Classification_scheme	UDK
* Classification_identifier	821.163.41-93
Description	
* Source	Autobiografija poznatog srpskog pisca Branislava Nušića ispričana u formi šaljivog romana. Sadrži i kraća objašnjenja manje poznatih reči.
* Language of Description	sr
Description	
* Source	Autobiography of famous Serbian writer Branisav Nušić but written as a funny story. Vocabulary sections included.
* Language of Description	en
Date and provenance	
* Date of origin	2001
* Provenance of origin	Belgrade
* Version of the cultural monument	First issue
Dimension	
* Dimension name	Number of pages
* Dimension value	283
Dimension	
* Dimension name	Book back height
* Dimension value	24 cm
Type	text
Acquisition	
* Type of acquisition	Legal deposit
* Date of acquisition	06/2001
Owner	NBS
Source object ID	II449580
Administrative	
Rights	Publishing house Kreativni centar

Access rights	Unlimited
Record creation date	30/01/2007
Record creator	Tamara Butigan Vučaj
Record owner	NBS
Extensions for librarian edition	
Publisher	Kreativni centar
Language	sr
Edition	Collection Pustolovine
Identifier	ISBN 86-7781-042-0

Table 5.4: Description of the object Digitalized-asset

5.2 Recommendations for the National Standard for Describing Collections

The metadata area is not resistant to Web 2.0 influences, on the contrary, it has been radically changing. Changes like social tagging, indexing and annotation bring more democracy in this traditionally conservative domain, including broader community in the kingdom of memory institutions. That is why it is important to have the technical environment supporting the aggregation of all available metadata and providing an open space for exchanging metadata. In this context, the NCD recommendations for metadata schemata are just the starting point to describe the heritage, leaving to the users to adjust it according their own needs.

This schema contains two groups of metadata elements:

- descriptive (17 elements) and
- administrative (5 elements),

listed below (mandatory fields are marked with asterisks, while ... denotes that the corresponding fields can be repeated):

- descriptive:
  - title*
    - original title*
    - title version ...
      - title version language*
      - title version*
  - creator ...
    - identifier
    - role ...
  - contributor ...
    - identifier

        - ∗ role ...
    - ◦ owner
    - ◦ subject ...
    - ◦ classification ...
        - ∗ classification scheme*
        - ∗ classification identifier*
    - ◦ description ...
        - ∗ source language
        - ∗ source
    - ◦ period of existence ...
        - ∗ date of creation
        - ∗ date of dismisson
        - ∗ comment ...
            - ⋆ source language
            - ⋆ source
    - ◦ coverage
    - ◦ type*
    - ◦ nature of collection
    - ◦ identifier*
    - ◦ collection's object ...
    - ◦ history ...
        - ∗ source language
        - ∗ source
    - ◦ related collection ...
        - ∗ type of relation*
        - ∗ identifier*
    - ◦ source object id
    - ◦ bibliography ...
    - ◦ note ...
        - ∗ source language
        - ∗ source
- administrative:
    - ◦ rights
    - ◦ access rights
    - ◦ record creation date
    - ◦ record creator
    - ◦ record owner

Example 5.2.1. *In this example it is illustrated how to describe the collection of the Serbian Children's digital library [73]. The author of the metadata records is Tamara Butigan Vučaj.*

Table 5.5 contains a record which corresponds to the description of this collection.

Descriptive	
Title	
Original Title	Srpska digitalna dečija biblioteka
* Version of Title * Language of Version	Serbian Children's digital library en
Owner	NBS
Type	Collection of Books
Nature of Collection	physical and virtual
Rights	NBS gained rights from the publishing houses
Access rights	Unlimited
Record creation date	21/4/2010
Record creator	Tamara Butigan Vučaj
Record owner	NBS

Table 5.5: Description of the Collection

5.3 Application

As it is said at the begging of this Chapter, Distributed Catalog of Serbian Digitized Cultural Collections is based on the Recommendations for the National Standard for Describing Collections and the Chord protocol. Web part of the application is active at `http://digitalne_kolekcije.ncd.org.rs/web_search/Search_jsp.jsp`.

5.3.1 Principles

One of the main features of a distributed catalog of digitized collections in Serbia is to assist researchers and members of the wider community in retrieving information concerning some fact of interest to them, information which can be provided from different kinds of sources. As mentioned before, digitized documents and collections, by their nature, are highly distributed resources. By connecting different kinds of data providers into one system, the quality of the resulting information can be increased.

This realization of the catalog contains only metadata on digital collections which follows a part of the Recommendation for the metadata format for describing collections, described in Section 5.2. One of the main reasons for this is the intellectual property rights issue. Simply, some institutions do not wish to outsource control over their digital repositories, and, instead, choose only to publish information about certain collections or some documents which are part of those collections.

A user can connect to one or more communities of which he is a member (i.e. he has been invited to or his request has been accepted). Two operations are then available, namely (i) storing a new record and (ii) finding a record which contains some information.

Suppose that someone wishes to store the following information on one digitized collection:

```
<collection>
  <title>
    <originalTitle>Title</originalTitle>
    <version lang="Language">Version</version>
  </title>
  <creator>
    <id>ID</id>
    <role>Role</role>
  </creator>
  <owner>Owner</owner>
  <type>Type</type>
  <natureOfCollection>Nature</natureOfCollection>
  <identifier>URL</identifier>
</collection>
```

To make the catalog searchable for the values in the fields: *originalTitle*, *version*, *creator*, *owner* and *natureOfCollection*, then the segments in accordance with Table 5.6 have to be stored. More precisely:

No.	Key	Value
1	$originalTitle\sharp Title$	$hash(✠)$
2	$version\sharp Version$	$hash(✠)$
3	$creator\sharp ID$	$hash(✠)$
4	$owner\sharp Owner$	$hash(✠)$
5	$natureOfCollection\sharp Nature$	$hash(✠)$
6	$hash(✠)$	✠

where ✠ represents the full meta-data record on one digital document

Table 5.6: Different data structures stored in the distributed catalog DHT for each entry

1. For every field of a meta-data record which is searchable, the hashed value for the current overlay of the entire meta-data record as value is stored with the key which contains information about the field and its value. Rows 1 to 5 in table 5.6.
2. The entire meta-data record as a value is stored with the corresponding key that contains its hashed value for the current overlay. Row 6 in table 5.6.

Note that all of the keys are stored with their hashed values. With this in place, the search mechanism has two phases. During the first phase, the hashed value of the meta-data record (the first kind of entries) will be found and then, during the second phase, the entire meta-data record (the second kind of entries) is looking for, but only in the overlays which contain the first kind of entries. Although the multiple copies of data exists, so as to accomplish failure resistance of the system, the storage space is of the same complexity as for a standard DBMS with indices. If N and M are the number of overlays and the number of nodes per overlay, respectively, then the time complexity of a search, in the worst case, is $O((N+1) * (\textit{time to search an overlay with } M \textit{ nodes}))$.

5.3.2 Description of the Distributed Catalog

Distributed catalog of digitized collections consists of two parts:

1. Web oriented part, and
2. “stand alone" Java application.

First part (Figure 5.2) is dedicated to the general public for searching. It is realized as a Java Servlet.

Figure 5.2: Distributed Catalog - Search Form

The results are then presented as clear HTML page (Figure 5.3).


Kolekcija: idvalue6
Originalni naziv: Virtuelna biblioteka Matematičkog fakulteta
Verzija naziva(eng): Virtual library of the Faculty of Mathematics
Autor:
Identifikator: Žarko Mijajlović
Uloga: author
Vlasnik kolekcije: Matematički fakultet Univerziteta u Beogradu
Opis(eng): The overall aim of our digital library is to create a comprehensive and interconnected collection of retro-digitized books and other digital documents. Objectives of the project include electronic archiving first of all the old manuscripts and their publishing in electronic photo-type form and presentation to the general audience. Our proposal was at the beginning mathematically inclined, but since 2009 we started to include the books from other areas of science and literature. However, the preference will be still given to Serbian authors and works related to South-East European (Balkan) area. Some of the books in our library are rather rare and it is known that there are left only few copies of them in the printed form. We think that it is very important to preserve them in some form. Not only as a cultural or scientific heritage important for our local community but also as a part of the World heritage. We decided to preserve and present them to the wide audience in the electronic - digitized form. Beside rare books, the library contains several important collections, for example doctoral dissertations of Serbian mathematicians (more than 350) and collected works of some leading Serbian scientists of the past: Bogdan Gavrilović, Milutin Milanković, Đuro Kurepa and few others. The library is continiously populated by new copies of digitized books. Papers related to this digital library are published in NCD Review. The project was started in 2007 by several volunteers. Now it has the strong support by the Faculty of Mathematics of the Belgrade University, Mathematical Institute of the Serbian Academy of Science and Art, Serbian Ministry of Science through the projects 13017 and III044006 and the National Center for Digitization. Since April 2010 the functionality of the library is based on DSpace, the software for building open digital repositories.
Datum kreiranja: 2010/4/
Tip: Scientific books and Articles
Priroda kolekcije: digital
Identifikator: http://elibrary.matf.bg.ac.rs/
Autorska prava: Public
Prava pristupa: Public
Kreator zapisa: Bojan Marinković
Vlasnik zapisa: Nacionalni centar za digitalizaciju

Figure 5.3: Distributed Catalog - Displaying Results

The “stand alone" Java application is dedicated to the system administrators and data providers. After connecting to the Catalog (Figure 5.4), the form for entering new records or editing existing ones appears (Figure 5.5).

With this application it is also possible to search the Catalog (Figure 5.6).

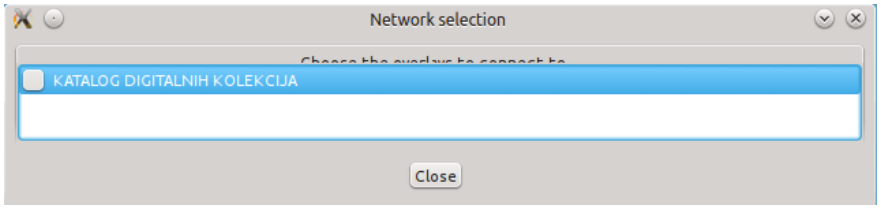


Figure 5.4: Distributed Catalog - Connecting Dialog

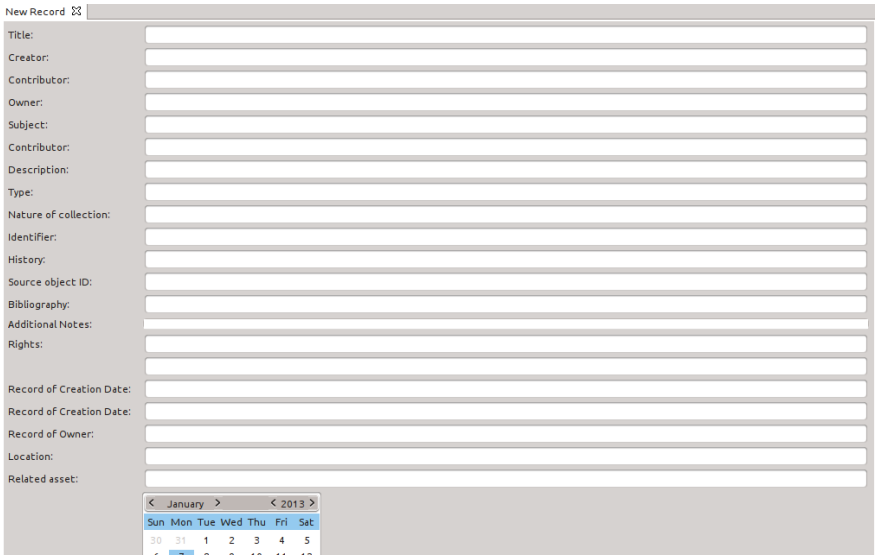


Figure 5.5: Distributed Catalog - Form for entering new/editing existing records

Figure 5.6: Distributed Catalog - Searching Dialog

6

Conclusions and Further Work

6.1 Conclusions

In this thesis an ASM-based formalization of the Chord protocol has been presented. It has been proved that the proposed formalization is correct with respect to the regular runs. This is the first comprehensive formal analysis of Chord presented in the literature which concerns both maintenance of the ring topology and data distribution. It has also been indicated that if all possible runs are considered, incorrect behavior of Chord protocol could appear. This proof has, also, been given in the refined form, where the temporal logic with time flow isomorphic to ω^2 has been introduced and the new semantics has been used with the low level description of the rules of the Chord protocol.

Then this ASM description of the Chord protocol is extended to give the specification of the Synapse protocol using the formalism of ASM and probabilistic assessments of the exhaustiveness of this protocol under a variety of scenarios has been given. It can be concluded that the exhaustiveness equations that have been proven are in strong correspondence with the results obtained by running the appropriate simulations and experiments. These equations have shown us that good exhaustiveness can be reached with a relatively small percentage of strategically positioned synapse nodes. It is always better to have a higher degree of connectivity of the synapses and an unlimited TTL, but even with relatively small numbers, reasonably good exhaustiveness can be achieved. To sum up, Synapse architectures can achieve a significant exhaustiveness using TTL, while reducing the expected overhead of communications and latency. Also, the granularity does not significantly influence communications when the number and connectivity of the synapses are fixed.

The scalability of the Synapse protocol and its applicability to a real-life situation has been proven.

6.2 Further Work

One possible direction for further work would be to apply technique presented in Chapter 3.1 to describe other DHT, Wireless and Ad hoc network protocols.

Another challenge could be verification of the given description in one of the formal proof assistants (e.g., Coq, Isabelle/HOL). It might also produce a certified program implementation from the proof of correctness of our ASM-based specification.

It would be possible to continue work on the full adoption of the semantics of temporal logic introduced in Section 3.2 in ASM inference system.

Also, the technique presented in Section 4.3 can be used for solving some open problems in the field of random multigraphs, especially when some parts of those graphs are not fully random but very-well structured.

The Synapse protocol has good potential as a new concept of DDBMS. With this concept, it is possible to connect heterogeneous DHTs in a homogeneous way. Since the full exhaustiveness of information retrieval cannot be guaranteed, the procedure of removing/updating items currently was out of scope of this research. The reason for this is that the only one who may remove or update items inside the catalog should be the one who inserted them in the first place, thus guaranteeing the highest probability of data consistency. For this, a User Management System has to be implemented, for instance, by implementing cryptographic techniques into the presented system, as described in [4]. Within this system the digital documents can be stored. The decision was, that this should also stay out of the scope of this thesis, but as a possible research in the future. As a positive side-effect, this catalog can lay as a promising groundwork for a low-cost solution to cultural interconnection of the institutions inside the region.

References

[1] Prior A. *Time and Modality.* Oxford University Press, 1957.

[2] N. Antonopoulos, J. Salter, and R. Peel. A Multi-Ring Method for Efficient Multi-Dimensional Data Lookup in P2P Networks. In *Proceedings of the 1st international conference on scalable information systems*, 2006.

[3] The Art & Architecture Thesaurus, AAT. `http://www.getty.edu/research/tools/vocabularies/aat/index.html`.

[4] A. Avramidis, P. Kotzanikolaou, and C. Douligeris. *Public Key Infrastructure*, chapter Chord-PKI: Embedding a Public Key Infrastructure into the Chord Overlay Network. Springer, 2007.

[5] R. Bakhshi and D. Gurov. Verification of Peer-to-Peer Algorithms: A Case Study. Technical report, ICT, 2006.

[6] R. Bakhshi and D. Gurov. Verification of Peer-to-Peer Algorithms: A Case Study. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 181:35–47, 2007.

[7] J. Barwise. *Admissible Sets and Structures.* Springer, 1975.

[8] G. Bella and E. Riccobene. Formal Analysis of the Kerberos Authentication System. *Journal of Universal Computer Science*, 3(12):1337–1381, 1997.

[9] M. Ben-Ari, A. Pnueli, and Z. Manna. The Temporal Logic of Branching Time. *Acta Informatica*, 20(3):207–226, 1983.

[10] E. Börger, Y. Gurevich, and D. Rosenzweig. The Bakery Algorithm: Yet Another Specification And Verification. *Specification and Validation Methods*, pages 231–243, 1995.

[11] E. Börger and A. Prinz. Quo Vadis Abstract State Machines? *Journal of Universal Computer Science*, 14(12):1921–1928, 2008.

[12] E. Börger and R. Stärk. *Abstract State Machines a Method for High-Level System Design and Analysis.* Springer-Verlag, 2003.

[13] D. Borsetti, C. Casetti, C. Chiasserini, and L. Liquori. *Heterogeneous Wireless Access Networks: Architectures and Protocols*, chapter Content Discovery in Heterogeneous Mobile Networks. Springer-Verlag, 2009.

[14] M. Botinčan, P. Glavan, and D. Runje. Distributed Algorithms. A Case Study of the Java Memory Model. In *Proc. of the 14th Int. ASM Workshop*, 2007.

[15] M. Botinčan, P. Glavan, and D. Runje. Verification of Causality Requirements in Java Memory Model is Undecidable PPAM. In *Parallel Processing and Applied Mathematics 8th International Conference*, volume 6068 of *LNCS*, pages 62–67, 2010.

[16] I. Buonazia, M. E. Masci, and D. Merlitti. The Project of the Italian Culture Portal and its Development. A Case Study: Designing a Dublin Core Application Profile for Interoperability and Open Distribution of Cultural Contents. In L. Chan and B. Martens, editors, *Proceedings of the 11th International Conference on Electronic Publishing ELPUB2007*, pages 393–494, 2007.

[17] J. Burgess. Logic and Time. *The Journal of Symbolic Logic*, 44(4):566–582, 1979.

[18] J. Burgess. Decidability for Branching Time. *Studia Logica*, 39(2-3):213–218, 1980.

[19] J. Burgess. Axioms for Tense Logic. I. “Since" and “Until". *Notre Dame Journal of Formal Logic*, 23(4):367–374, 1982.

[20] J. Burgess. Basic tense logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, pages 89–133. Heidel Publishing Company, 1984.

[21] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *Proceedings of the* 7^{th} *Conference on Usenix Symposium on Operating Systems Design and Implementation*, volume 7, pages 205–218, 2006.

[22] L. Cheng. Bridging Distributed Hash Tables in Wireless Ad-Hoc Networks. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 5159–5163, 2007.

[23] L. Cheng, R. Ocampo, K. Jean, A. Galis, C. Simon, R. Szabo, P. Kersch, and R. Giaffreda. Towards Distributed Hash Tables (De) Composition in Ambient Networks. In *LNCS*, volume 4269, 2006.

[24] V. Ciancaglini, L. Liquori, and L. Vanni. CarPal: Interconnecting Overlay Networks for a Community-Driven Shared Mobility. In *Trustworthy Global Computing*, pages 301–317, 2010.

[25] A. Datta and K. Aberer. The Challenges of Merging Two Similar Structured Overlays: A Tale of Two Networks. In *IWSOS*, 2006.

[26] S. Demri and D. Nowak. Reasoning about Transfinite Sequences. *International Journal of Foundations of Computer Science*, 18(1):87–112, 2007.

[27] S. Demri and A. Rabinovich. The Complexity of Temporal Logic with Until and Since over Ordinals. *Logical Methods in Computer Science*, 6(4), 2010.

[28] Digital National Library of Serbia. http://www.digitalna.nb.rs/.

[29] Distributed and Mobile Systems Group Lehrstuhl für Praktische Informatik Universität Bamberg. open-chord v. 1.0.5 implementation, 2008.

[30] D. Doder, Z. Marković, Z. Ognjanović, A. Perović, and M. Rašković. A Probabilistic Temporal Logic that Can Model Reasoning about Evidence. In *LNCS*, volume 5956, pages 9–24, 2010.

[31] D. Doder, Z. Ognjanović, and Z. Marković. An Axiomatization of a First-Order Branching Time Temporal Logic. *Journal of Universal Computer Science*, 16(11):1439–1451, 2010.

[32] The Dublin Core Metadata Initiative. http://dublincore.org/.

[33] E. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Sematics, pages 995–1072. North-Holland Pub. Co./MIT Press, 1990.

[34] E. Emerson and E. Clarke. Using Branching Time Logic to Synthesize Synchronization Skeletons. *Sci. Comput. Program.*, 2:241–266, 1982.

[35] E. Emerson and J. Halpern. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. *Journal of Computer and Systems Science*, 30(1):1–24, 1985.

[36] Encoded Archival Description. http://www.loc.gov/ead/.

[37] L. G. Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. U. Keller. Hierarchical P2P Systems. In *Euro-Par*, 2003.

[38] The European Library. http://www.theeuropeanlibrary.org/.

[39] Europeana. http://www.europeana.eu.

[40] Y. Feldman. A Decidable Propositional Dynamic Logic with Explicit Probabilities. *Information and Control*, 63:11–38, 1984.

[41] L. Ferrucci, D. Mandrioli, A. Morzenti, and M. Rossi. A Metric Temporal Logic for Dealing with Zero-Time Transitions. In *International Syposium on Temporal Representation and Reasoning*, pages 81–88, 2012.

[42] P. Furtado. Multiple Dynamic Overlay Communities and Inter-Space Routing. In *LNCS*, volume 4125, 2007.

[43] D. Gabbay, I. Hodkinson, and M. Reynolds. Temporal Logic. *Mathematical Foundations and Computational Aspects*, 1, 1994.

[44] A. Gargantini, D. Mandrioli, and A. Morzenti. Dealing with Zero-Time Transitions in Axiom Systems. *Information and Computation*, 150(2):119–131, 1999.

[45] Y. Gurevich. Evolving Algebras 1993: Lipari Guide. *Specification and Validation Methods*, pages 9–36, 1995.

[46] Y. Gurevich. Sequential Abstract State Machines Capture Sequential Algorithms. *ACM Transactions on Computational Logic*, 1(1):77–111, 2000.

[47] Y. Gurevich and J. K. Huggins. The Railroad Crossing Problem: An Experiment with Instantaneous Actions and Immediate Reactions. In *Computer Science Logic, Selected papers from CSL'95*, volume 1092 of *LNCS*, pages 266–290, 1996.

[48] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspect of Computing*, 6(5):512–535, 1994.

[49] IFLA Study Group, Functional Requirements for Bibliographic Records. `http://www.ifla.org/VII/s13/frbr/frbr.pdf`, 1998.

[50] K. Junjiro, W. Naoki, and M. Masayuki. Design and Evaluation of a Cooperative Mechanism for Pure P2P File-Sharing Networks. In *IEICE Trans Commun (Inst Electron Inf Commun Eng)*, pages 2319–2326, 2006.

[51] D.R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, and D. Lewin. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proceedings of STOC'97*, pages 654–663, 1997.

[52] S. Krishnamurthy, S. El-Ansary, E. Aurell, and S. Haridi. A Statistical Theory of Chord Under Churn. In *4th International Workshop on Peer-To-Peer Systems*, pages 93–103, 2005.

[53] M. Kwon and S. Fahmy. Synergy: An Overlay Internetworking Architecture. In *Proc. of International Conference on Computer Communications and Networks*, pages 401–406, 2005.

[54] A. Lakshman and P. Malik. Cassandra - A Decentralized Structured Storage System. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[55] D. Lehmann and S. Shelah. Reasoning with Time and Chance. *Information and Control*, 53:165–198, 1982.

[56] D. Liben-Nowell, H. Balakrishnan, and D. R. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proc.* 21^{st} *ACM Symp. Principles of Distributed Computing (PODC)*, pages 233–242, 2002.

[57] L. Liquori, D. Borsetti, C. Casetti, and C. Chiasserini. An Overlay Architecture for Vehicular Networks. In *IFIP Networking, International Conference on Networking*, volume 4982 of *LNCS*, pages 60–71. Springer, 2008.

[58] L. Liquori, C. Tedeschi, and F. Bongiovanni. BabelChord: A Social Tower of DHT-Based Overlay Networks. In *14th Symposium on Computers and Communications (ISCC 2009) IEEE*, 2009. Short paper.

[59] L. Liquori, C. Tedeschi, L. Vanni, F. Bongiovanni, V. Ciancaglini, and B. Marinković. Synapse: A Scalable Protocol for Interconnecting Heterogeneous Overlay Networks. In *Networking 2010*, volume 6091 of *LNCS*, pages 67–82, 2010.

[60] E. J-L. Lu, Y-F. Huang, and S-C. Lu. ML-Chord: A Multi-Layered P2P Resource Sharing Model. *Journal of Network and Computer Applications*, 32:578–588, 2009.

[61] Reynolds. M. An Axiomatization of Full Computation Tree Logic. *The Journal of Symbolic Logic*, 66(3):1011–1057, 2001.

[62] MARC Standards. `http://www.loc.gov/marc/`.

[63] B. Marinković. Applying XML Technologies for Digitization of the National Cultural Heritage. Master's thesis, The Faculty of Mathematics University of Belgrade, 2008.

[64] B. Marinković, L. Liquori, V. Ciancaglini, and Z. Ognjanović. A Distributed Catalog for Digitized Cultural Heritage. In *ICT Innovations 2010*, volume 83 of *CCIS*, pages 176–186, 2011.

[65] MICHAEL Collection Description. `http://www.mla.gov.uk/what/publications/~/media/Files/pdf/2007/michael_manual_v2.ashx`.

[66] OAIster. `http://www.oclc.org/oaister/`.

[67] Zoran Ognjanović. Discrete Linear-Time Probabilistic Logics: Completeness, Decidability and Complexity. *Journal of Logic Computation*, 16(2):257–285, 2006.

[68] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th IEEE Symposium Foundations of Computer Science (FOCS 1977)*, pages 46–57, 1977.

[69] Recommendation for the National Standard for Describing Digitized Heritage in Serbia. `http://www.ncd.org.rs/ncd_sr/standards/ncd_dokumenti_2_0.html`.

[70] Recommendations for Coordination of Digitization of Cultural Heritage in South-Eastern Europe. Review of the National Center for Digitization, `http://elib.mi.sanu.ac.rs/files/journals/ncd/7/ncd07002.pdf`, 2005.

[71] L. Rideau, B. Serpette, and C. Tedeschi. Formalizaton and Concretization of Ordered Networks. Technical report, INRIA, 2012.

[72] R. Rodrigues and P. Druschel. Peer-to-Peer Systems. *Communications of the ACM*, 53(10):72–82, October 2010.

[73] Serbian Children's Digital Library. `http://www.digitalna.nb.rs/wb/NBS/Knjige/Srpska_decja_digitalna_biblioteka/`.

[74] T. M. Shafaat, A. Ghodsi, and S. Haridi. Handling Network Partitions and Mergers in Structured Overlay Networks. In *Proc. of P2P*, pages 132–139. IEEE Computer Society, 2007.

[75] A. Sistla and E. Clarke. The Complexity of Propositional Linear Temporal Logic. *JACM*, 32(3):733–749, 1985.

[76] SPECTRUM, the UK Museum Documentation Standard. `http://www.mda.org.uk/spectrum.htm`.

[77] C. Stirling. *Handbook of Logic in Computer Science*, volume 2, chapter Modal and Temporal Logic, pages 477–563. Oxford University Press, 1992.

[78] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM*, pages 149–160, 2001.

[79] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. Technical report, MIT, 2001.

[80] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.

[81] I. Taylor. *From P2P to Web Services and Grids*. Springer-Verlag, 2005.

[82] V. Tru'o'ng. Testing Implementations of Distributed Hash Tables. Master's thesis, IT Univesity of Göteborg, 2007.

[83] A Weblog about FRBR: Functional Requirements for Bibliographic Records. `http://www.frbr.org/`, 2007.

[84] Z. Xu, R. Min, and Hu. Y. HIERAS: A DHT Based Hierarchical P2P Routing Algorithm. In *ICPP*, 2003.

[85] P. Zave. Using Lightweight Modeling to Understand Chord. *ACM SIGCOMM Computer Communication Review*, 42(2):50–57, April 2012.

List of Figures

List of Tables

Appendix A

Appendix: Chord Rules - Low Level Description

Here, a detailed specification of the module and rules introduced in Section 3.1.2 will be presented. The lines on the right side given in the different font correspond to the lines from the high level description.

```
Main Module=
// Peer tries to start/connect to the Chord network
if mode(Me) = not_connected then
  if mode_join(Me) = undef then
    if action = undef then
      // Peer has been not active and can choose to skip or to join
      choose a in Join                          Chosen Action Is Join
        action := a
      endchoose
    else
      if action = join then
        seq
          // Checking the list of known nodes of Chord network
          known := known_nodes(Me)
          if known = undef then                 There Are No Known Nodes
            // Start new Chord network
            Start
          else
            // Join to existing Chord network
            Join =
            if mode_join(Me) = undef then
              JoinBegin
            else
              // Continuing joining process depending on received mes-
sages
              if mode_join(Me) = wait_for_successor then
                JoinGetSuccessor
              else
                if mode_join(Me) = wait_for_keys then
```

```
                    JoinGetKeys
                  else
                    par
                      // Connection was successful - start stabilization
process
                      if id(Me) ≠ undef then
                        mode(Me) := connected              Connection
Successful
                      // Connection was not successful - try again
                      else
                        mode(Me) := not_connected
                      endif
                      mode_join(Me) = undef
                      action := undef
                    endpar
                  endif
                endif
              endif
            endif
          endseq
        else
          // the skip action has been chosen
          action := undef
        endif
    endif
// Connected node chooses what to do
else
    if mode(Me) = connected then          Does Not Have Communication
Problems
      // Checking if id(Me) can communicate with other nodes
      if ping(id(Me)) = true then
        par
          // Read all new messages
          ReadMessages
          // Begin new stabilization cicle
          Stabilize =
          if mode_stabilize(Me) = undef
              ∧(action ≠ fair_leave ∨ action ≠ unfair_leave) then
            StabilizeBegin
          else
            if mode_stabilize(Me) = wait_for_predecessor then
              StabilizeWithPredecessor
            else
              if mode_stabilize(Me) = wait_for_keys then
                StabilizeSetKeys
              else
                if mode_stabilize(Me) = wait_for_successor then
                  StabilizeUpdateSuccessor
                else
                  if mode_stabilize(Me) = wait_for_successor_keys then
```

```
            StabilizeUpdateKeys
          endif
        endif
      endif
    endif
endif
// Cheking connectivity of the predecessor
UpdatePredecessor
// Update value for a finger table member
UpdateFingers =
if mode_fingers(Me) = undef
    ∧(action ≠ fair_leave ∨ action ≠ unfair_leave) then
  UpdateFingersBegin
else
  if mode_fingers(Me) = wait_for_response
    UpdateFingersSet
  endif
endif
// Allow other actions beside Join
ExtendedJoinModel =
seq
  if action = undef then
    choose a in Action
      action := a
    endchoose
  endif
  par
    // Allow leaving
    LeavingActions =
    par
      FairLeave =
      if action = fair_leave ∧ mode_stabilize(Me) = undef
          ∧mode_fingers(Me) = undef then
        // Node is leaving network in a fair way
        if mode_leave(Me) = undef then
          FairLeaveGetSuccessor
        else
          if mode_leave(Me) = wait_for_successor then
            FairLeaveUpdateSuccessor
          else
            par
              FairLeaveExchange
              mode(Me) := not_connected
              action := undef
            endpar
          endif
        endif
      endif
      if action = unfair_leave∧mode_stabilize(Me) = undef
          ∧mode_fingers(Me) = undef then
```

```
                par
                  // Node is leaving network in a unfair way
                  UnfairLeave
                  mode(Me) := not_connected
                  action := undef
                endpar
              endif
            endpar
            // Allow key/value handling
            KeyValueHandling =
            par
              Put =
              if action = put then
                // Inserting new ⟨key, value⟩ pair
                if mode_put(Me) = undef then
                  seq
                    ⟨key, value⟩ := key_value(Me)
                    PutFindResponsible
                  endseq
                else
                  par
                    PutKeyValue
                    action := undef
                  endpar
                endif
              endif
              Get =
              if action = get then
                if mode_get(Me) = undef then
                  seq
                    // Searching for existing value
                    key := keys(Me)
                    GetKey
                  endseq
                else
                  if mode_get(Me) = wait_for_key then
                    GetValue
                  else
                    par
                      GetFinish
                      action := undef
                    endpar
                  endif
                endif
              endif
            endpar
          endpar
        endseq
      endpar
    else
```

```
        // Connection problems detected - reset connection
        par
          mode(Me) := not_connected
          mode_stabilize(Me) := undef
          mode_fingers(Me) := undef
          mode_leave(Me) := undef
          mode_put(Me) := undef
          mode_get(Me) := undef
          action := undef
        endpar
      endif
    endif
  endif

Start=
seq
  // Setting id(Me) according to the hash function
  id(Me) := hash(Me)
  // Set all messages to id(Me) to empty
  ClearMessages
  // Initialization of local structure          Initialize Values For Node
  par
    predecessor(id(Me)) := undef
    successor(id(Me)) := id(Me)
    finger(id(Me)) := []
    next(id(Me)) := -1
    keyvalue(id(Me)) := []
  endpar
endseq

JoinBegin=
seq
  // Setting id(Me) according to the hash function
  id(Me) := hash(Me)
  // If connection is succesful
  if id(Me) ≠ undef then                      Chord Is Not Fulfilled
    seq
      // Set all messages to id(Me) to empty
      ClearMessages
      // Initialize local structure
      seq                                     Initialize Values For Node
        par
          predecessor(id(Me)) := undef
          finger(id(Me)) := []
          next(id(Me)) := -1
        endpar
        // Ask for the successor
        communication(id(Me), known).add(⟨find_successor, ⟨id(Me), id(Me)⟩⟩)
        mode_join(Me) := wait_for_successor
      endseq
```

```
    endseq
  else
    // Connection was not succesful
    mode_join(Me) := finished
  endif
endseq

JoinGetSuccessor=
// Get the message with the information on the successor
forall m ∈ Message with m = communication(sender, id(Me))
  if m = ⟨successor_found, content⟩ then
    if content = ⟨id(Me), successor⟩ then
      par
        // Set successor
        successor(id(Me)) := successor
        // Ask successor for keys                    Take keyvalue
        communication(id(Me), successor(id(Me))
          .add(⟨request_and_remove_keyvalue, undef⟩)
        mode_join(Me) := wait_for_keys
        // Remove processed message
        communication(sender, id(Me)).remove(m)
      endpar
    endif
  endif
endforall

JoinGetKeys=
// Get the message with keys and values
forall m ∈ Message with m = communication(sender, id(Me))
  if m = ⟨received_keyvalue, content⟩ then
    par
      // Set keys
      keyvalue(id(Me)) := content
      mode_join(Me) := finished
      // Remove processed message
      communication(sender, id(Me)).remove(m)
    endpar
  endif
endforall

FairLeaveUpdateSuccessor=
// Check the successor and update it if necessary
if ¬ping(successor(id(Me))) then      successor(id(Me)) Is Not Alive
  seq                                        Update successor(id(Me))
    communication(id(Me), id(Me)).add(⟨find_successor, ⟨id(Me)⊕N1, id(Me)⟩⟩)
    mode_leave(Me) := wait_for_successor
  endseq
else
  mode_leave(Me) := proceed_to_finish
endif
```

```
FairLeaveGetSuccessor=
seq
  // Get the message with the information on the successor
  forall m ∈ Message with m = communication(sender, id(Me))
    if m = ⟨successor_found, content⟩ then
      if content = ⟨id(Me), successor⟩ then
        par
          // Set successor
          successor(id(Me)) := successor
          // Remove processed message
          communication(sender, id(Me)).remove(m)
        endpar
      endif
    endif
  endforall
  mode_leave(Me) := proceed_to_finish
endseq

FairLeaveExchange=
seq
  // If Me is not the last node  id(Me) Is Not The Only Node In Chord
  if ¬(successor(id(Me)) = predecessor(id(Me)) ∧ predecessor(id(Me)) =
id(Me)) then
    par
      // Send keys/values to the successor          Give keyvalue(id(Me))
      communication(id(Me), successor(id(Me))).add(⟨send_keys, keyvalue(id(Me))⟩)
      // Exchange pointers between the successor and the predecessor
      communication(id(Me), successor(id(Me)))          Remove id(Me)
        .add(⟨send_predecessor, predecessor(id(Me))⟩)
      communication(id(Me), predecessor(id(Me)))
        .add(⟨send_successor, successor(id(Me))⟩)
      // Clean local memory                              Deactivate Values
      predecessor(id(Me)) := undef
      successor(id(Me)) := undef
      finger(id(Me)) := []
      keyvalue(id(Me)) := []
    endpar
  endif
endseq

StabilizeBegin=
// Check if id(Me) can communicate with other nodes
if ping(id(Me)) = true then
  // Check the successor's connection
  if ping(successor(id(Me))) = true then   successor(id(Me)) Is Alive
    par
      // Get information on the predecessor of the successor
      communication(id(Me), successor(id(Me))).add(⟨get_predecessor, undef⟩)
      mode_stabilize(Me) := wait_for_predecessor
    endpar
```

```
  else
    par
      // Find new successor
      communication(id(Me), id(Me)).add(⟨find_successor, ⟨id(Me)⊕N 1, id(Me)⟩⟩)
      mode_stabilize(Me) := wait_for_successor
    endpar
  endif
endif

StabilizeWithPredecessor=
// Get the message with the information on the predecessor of the successor
forall m ∈ Message with m = communication(sender, id(Me)) do
  if m = ⟨received_predecessor, content⟩ then
    seq
      x := content          Set x To Be predecessor(successor(id(Me)))
      // Remove processed message
      communication(sender, id(Me)).remove(m)
      // There is a new node between id(Me) and the successor
      if (x ≠ undef ∧ member_of(x, id(Me), successor(id(Me)))) then
        par
          // Update the successor and take keys/values from it
          successor(id(Me)) := x
          communication(id(Me), successor(id(Me))      Take Keyvalue
            .add(⟨request_and_remove_keyvalue, undef⟩)
          mode_stabilize(Me) := wait_for_keys
        endpar
      endif
      // id(Me) is a new node between successor and its predecessor
      if x = undef∨(x ≠ undef∧member_of(id(Me), x, successor(id(Me)))) then
          // Notify the successor to change the predecessor
          par   Notify successor(id(Me)) To Update Its predecessor
            communication(id(Me), successor(id(Me))).add(⟨set_predecessor, undef⟩)
            mode_stabilize(Me) := undef
          endpar
      endif
    endseq
  endif
endforall

StabilizeSetKeys=
// Get the message with keys and values
forall m ∈ Message with m = communication(sender, id(Me))
  if m = ⟨received_keyvalue, content⟩ then
    par
      // Add keys/values to local table
      keyvalue(id(Me).add(content)
      mode_stabile(Me) := undef
      // Remove processed message
      communication(sender, id(Me)).remove(m)
    endpar
```

```
  endif
endforall

StabilizeWaitSuccessor=
// Get the message with the information on the successor
forall m ∈ Message with m = communication(sender, id(Me))
  if m = ⟨successor_found, content⟩ then
    if content = ⟨id(Me) ⊕_N 1, successor⟩ then
      par
        // Set successor
        successor(id(Me)) := successor
        // Ask successor for keys
        communication(id(Me), successor(id(Me))
          .add(⟨request_and_remove_keyvalue, undef⟩)
        mode_stabilize(Me) := wait_for_successor_keys
        // Remove processed message
        communication(sender, id(Me)).remove(m)
      endpar
    endif
  endif
endforall

StabilizeUpdateKeys=
// Get the message with keys and values
forall m ∈ Message with m = communication(sender, id(Me))
  if m = ⟨received_keyvalue, content⟩ then
    par
      // Add keys/values to local table
      keyvalue(id(Me).add(content)
      mode_stabilize(Me) := undef
      // Remove processed message
      communication(sender, id(Me)).remove(m)
    endpar
  endif
endforall

UpdatePredecessor=
// Check if id(Me) can communicate with other nodes
if ping(id(Me)) = true then
  // Check the predecessor's connection
  if ping(predecessor(id(Me))) ≠ true then Predecessor Is Not Alive
    predecessor(id(Me)) := undef                    Deactivate Value
  endif
endif

UpdateFingersBegin=
// Update a finger table item
par
  next(id(Me)) := (next(id(Me)) ⊕_M 1) + 1
  communication(id(Me), id(Me)).add(⟨find_successor, ⟨id(Me)⊕_N 2^(next(id(Me))−1), id(Me)⟩⟩)
```

```
   mode_fingers(Me) := wait_for_response
endpar

UpdateFingersSet=
// Get the message with the information on the successor
forall m ∈ Message with m = communication(sender, id(Me))
   if m = ⟨successor_found, content⟩ then
      if content = ⟨id(Me) ⊕_N 2^(next(id(Me))−1), successor⟩ then
         par
            // Set current finger table item
            finger(id(Me)).listitem(next(id(Me))) := successor
            // Remove processed message
            communication(sender, id(Me)).remove(m)
            //  Begin update next table item
            mode_fingers(Me) := undef
         endpar
      endif
   endif
endforall

PutFindResponsible=
// Insert new ⟨key, value⟩ pair into Chord network
par
   // Find the responsible node for hash(key)
   communication(id(Me), id(Me)).add(⟨find_successor, ⟨hash(key), id(Me)⟩⟩)
   mode_put(Me) := wait_for_responsible
endpar

PutKeyValue=
// Get the message with the information on the responsible node
forall m ∈ Message with m = communication(sender, id(Me))
   if m = ⟨successor_found, content⟩ then
      if content = ⟨hash(key), successor⟩ then
         par
            x := successor
            if x ≠ undef then
               // Send ⟨key, value⟩ to the responsible node Notify Responsible
Node
               communication(id(Me), x).add(⟨send_keys, [⟨hash(key), value⟩]⟩)
            else
               Skip
            endif
            // Remove processed message
            communication(sender, id(Me)).remove(m)
            // Finishing this instering operation
            mode_put(Me) := undef
         endpar
      endif
   endif
endforall
```

```
GetKey=
// Begin search for given value
par                                                    Invoke FindSuccessor
  communication(id(Me), id(Me)).add(⟨find_successor, ⟨hash(key), id(Me)⟩⟩)
  mode_get(Me) := wait_for_key
endpar

GetValue=
// Get the message with the information on the responsible node
forall m ∈ Message with m = communication(sender, id(Me))
  if m = ⟨successor_found, content⟩ then
    if content = ⟨hash(key), successor⟩ then
      par
        // Ask responsible node for the value connected with the given key
        responsible := successor                    Check Corresponding value
        communication(id(Me), responsible).add(⟨get, hash(key)⟩)
        // Remove processed message
        communication(sender, id(Me)).remove(m)
        mode_get(Me) := wait_for_value
      endpar
    endif
  endif
endforall

GetFinish=
// Get the message with the information on the asked value
forall m ∈ Message with m = communication(sender, id(Me))
  if m = ⟨value_found, content⟩ then
    if content = ⟨hash(key), successor⟩ then
      par
        value := content
        // Remove processed message
        communication(sender, id(Me)).remove(m)
        mode_get(Me) := undef
      endpar
    endif
  endif
endforall

Search=
// Find given value in local key/value table
seq
  choose value in Value satisfying
    ⟨hash(key), value⟩ ∈ keyvalue(id(Me))
      content := value
  endchoose
  communication(id(Me), sender).add(value_found, content)
endseq

ClearMessages=
```

```
// Clear messages remaining by the node which had same id
forall node ∈ Chord
  communication(node, id(Me)) := []
endforall

ReadMessages=
// Process received messages
forall m ∈ Message with m = communication(sender, id(Me)) do
  seq
    par
    // Send proper keys/values to new predecessor
    if m = ⟨request_and_remove_keyvalue, content⟩ then
        seq
          res := []
          forall ⟨h, v⟩ ∈ keyvalue(id(Me)) with h ⩽ sender do
            par
              keyvalue(id(Me)).remove(⟨h, v⟩)
              res.add(⟨h, v⟩)
            endpar
          endforall
          communication(id(Me), sender) := ⟨received_keyvalue, res⟩
        endseq
      endif
      // Add received keys/values to local table
      if m = ⟨send_keys, content⟩ then
        keyvalue(id(Me)).add(content)
      endif
      // Set given predecessor
      if m = ⟨set_predecessor, content⟩ then
        predecessor(id(Me)) := sender
      endif
      // Send information on local predecessor
      if m = ⟨get_predecessor, content⟩ then
        communication(id(Me), sender) := ⟨received_predecessor, predecessor(id(Me))⟩
      endif
      // Process query on respoinsible node
      if m = ⟨find_successor, content⟩ then
        FindSuccessor
      endif
      // Process search on given value in local table
      if m = ⟨get, content⟩ then
        Search
      endif
    endpar
    // Remove processed message
    communication(sender, id(Me)).remove(m)
  endseq
endforall

FindSuccessor=
```

```
seq
  ⟨h, starter⟩ := content
  // If h is between id(Me) and its successor responsible node is the successor
  if ping(successor(id(Me)))∧member_of(h, id(Me), successor(id(Me))) then
                                                                    Respond
    communication(id(Me), starter).add(⟨successor_found, ⟨h, successor(id(Me))⟩⟩)
  else
  // Find the closest preceding node in local finger table
    seq
      index := undef
      choose i in {1,...,M} satisfying
        member_of(finger(id(Me)).listitem(i), id(Me), h)
          ∧¬member_of(finger(id(Me)).listitem(i + 1), id(Me), h)))
            ∧ping(finger(id(Me)).listitem(i)) = true
            // h is between two values in local finger table
            index := i
      // h is bigger then all values in local finger table - the closest is the
maximal element
      if index = undef then
        choose i in {1,...,M} satisfying
          ping(finger(id(Me)).listitem(i)) = true
            ∧(∀j ∈ {2,...,M + 1|j > i})ping(j) = false
        index := i
      endif
      // Forward query to it                                  Forward Query
      communication(id(Me), finger(id(Me)).listitem(index))
        .add(⟨find_successor, ⟨h, starter⟩⟩)
    endseq
  endif
endseq
```

Biography

Bojan Marinković was born in Belgrade on January 11^{th}, 1980. He graduated on 2004 from the Faculty of Mathematics, University of Belgrade. Under the supervision of prof. Gordana Pavlović-Lažetić during 2008, he received MSc with “Applying XML Technologies for Digitization of the National Cultural Heritage". On 2008/2009, he became PhD student at The Faculty of Technical Sciences University of Novi Sad within the program “Mathematics in Technics".

He has been employed at the Mathematical Institute of the Serbian Academy of Sciences and Arts from January 1^{st}, 2006, firstly as Research Assistant Trainee, and after as Research Assistant. He has published several articles in leading journals and international conferences in the field of digitization of the cultural and scientific heritage, mathematical logic and overlay networks. He has participated in the realization of a large number of the projects of Mathematical Institute. He has participated in a large number of seminars and summer schools. During the realization of the Tempus project DEUKS he spent three months doing an internship at LogNet team of INRIA Sophia Antipolis. He was a member of the organizing committee of the conference RDP 2011.

Beside all this activities, he is administrator of the clusters of Mathematical Institute, and since April 2010 he has been working as Registration Authority Person for AEGIS11-MISANU GRID site.

Biografija

Bojan Marinković je rođen u Beogradu 11.01.1980. godine. Diplomirao je 2004. godine na Matematičkom fakultetu Univerziteta u Beogradu na smeru računarstvo i informatika. Pod mentorstvom Gordane Pavlović-Lažetić 2008. godine odbranio je master rad „Primena XML tehnologija u digitalizaciji nacionalne kulturne baštine". Školske 2008/2009 godine upisao je doktorske studije na Fakultetu tehničkih nauka Univerziteta u Novom Sadu na studijskom programu „Matematika u tehnici".

U Matematičkom institutu SANU zaposlen je od 01.01.2006. godine, prvo kao istraživač pripravnik, a zatim i kao istraživač saradnik. Objavio je veći broj radova u vodećim časopisima i na vodećim međunarodnim konferencijama iz oblasti digitalizacije kulturne i naučne baštine, matematičke logike i prekrivajućih mreža. Učestvovao je u realizaciji brojnih projekta Matematičkog instituta SANU. Polaznik je mnogobrojnih seminara i letnjih škola. Tokom realizacije Tempus projekta DEUKS proveo je tri meseca u LogNet timu istraživačkog centra INRIA Sofia Antipolis. Bio je član organizacionog komiteta međunarodne konferencije RDP 2011.

Pored ovoga, obavlja poslove administriranja klastera u Matematičkom institutu SANU, a od aprila 2010. godine je nadležan za registraciju korisnika za AEGIS11-MISANU GRID sajta.