Specification and Verification of Multi-Agent Systems

Course Reader

Wojciech Jamroga
Polish Academy of Sciences
wojciech.jamroga@ipipan.waw.pl

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Introduction

Overview. Multi agent systems (MAS) provide an important framework for formalizing various problems in computer science, artificial intelligence, game theory, social choice theory, etc. Modal logics are amongst the most suitable and versatile logical formalisms for specification and verification of computational systems. The course offers an introduction to some important developments in the area. We introduce modal logics used for specification of temporal, epistemic, and strategic properties of systems. Then, we present some model checking algorithms, and discuss the computational complexity of the model checking problem. We also consider symbolic (compact) representations of systems, and how the representations change the semantic and algorithmic side of model checking. Finally, we discuss some techniques that help to reduce the complexity and make verification feasible even for large systems.

This course is based on a postgraduate course given in 2010 by myself and Wojciech Penczek (at the ESSLLI summer school in Copenhagen). Moreover, the reader is based on the following book chapter:

> Wojciech Jamroga and Wojciech Penczek (2012), Specification and Verification of Multi-Agent Systems. In: *Lectures on Logic and Computation*, Lecture Notes in Computer Science vol. 7388, pp. 210-263. Springer.

I gratefully acknowledge the permission of my coauthor, Wojciech Penczek, to use his parts of the chapter in this reader, as well as his parts of the lecture slides.

Contents. The course consists of 3 main sections, with the content outlined below:

1. *Introduction to logic-based specification and verification of multi-agent systems.* Agents and agent systems. Why logic? Modal logic: modal operators, possible worlds semantics. Modal logic as a generic framework to model and reason about MAS. Basic multi-agent epistemic logic. Common knowledge and distributed knowledge. Decision problems. Formal verification by model checking. Some complexity classes. Model checking epistemic properties.

2. *Reasoning about evolution of systems.* Multi-agent systems. Modal logic, Kripke models, epistemic logic. Temporal logic, linear vs. branching time, μ-calculus. Basic complexity results. Combining modalities: reasoning about knowledge and time, interpreted systems, BDI logics.

3. *Introduction to model checking for knowledge and time.* Standard non-symbolic algorithms. Interleaved interpreted systems, partial order reduction methods. Introduction to symbolic model checking, OBDD-based approaches.

Further reading. The following publications may be of particular interest to students of this course:

- Wojciech Jamroga and Wojciech Penczek (2012), Specification and Verification of Multi-Agent Systems. In: *Lectures on Logic and Computation*, Lecture Notes in Computer Science vol. 7388, pp. 210-263. Springer.
- Alessio Lomuscio and Wojciech Penczek (2012), Symbolic Model Checking for Temporal-Epistemic Logic. In: *Logic Programs, Norms and Action*, Lecture Notes in Computer Science vol. 7360, pp. 172-195, Sptinger.
- Yoav Shoham and Kevin Leyton-Brown (2009), *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.
- Edmund M. Clarke, Orna Grumberg and Doron A. Peled (1999), *Model Checking*. MIT Press.

We give more detailed references for further reading throughout the rest of the materials.

Acknowledgements. The materials and slides include fragments that had been contributed by a number of people. Most of all, I would like to thank Wojciech Penczek for his input. Moreover, I gratefully acknowledge contributions of: Thomas Agotnes, Nils Buling, Jürgen Dix, Valentin Goranko, Hongyang Hu, Magdalena Kacprzak, Alessio Lomuscio, Maciej Szreter, and Bożena Woźna-Szcześniak.

About the lecturer. Wojciech Jamroga is an associate professor at the Polish Academy of Sciences, Poland, and a Privatdozent at the Clausthal University of Technology, Germany. He works mainly in the area of multi-agent systems. His research is focused on modal logics for reasoning about agents and game-theoretic analysis of MAS. He has been recently awarded a Marie Curie grant for the project ReVINK (Representation and Verification of Interaction and Knowledge), and moved back to Poland after 14 years of research work abroad.

Wojciech Jamroga obtained his PhD from the University of Twente (Netherlands) in 2004, and completed his habilitation at the Clausthal University of Technology (Germany) in 2009. He has coauthored over 50 refereed publications, and is a Program Committee member of the most important conferences and workshops in multi-agent systems. His teaching record includes multiple courses at ESSLLI (European Summer School in Logic, Language and Information), and EASSS (European Agent Systems Summer School), all of them on logical aspects of multi-agent systems.

Chapter 1

Logics for Multi-Agent Systems

This section serves as an introduction of some fundamental concepts that we are going to use throughout.

1.1 Multi-Agent Systems

Multi-agent systems (*MAS*) are systems that involve several autonomous entities acting in the same environment. The entities are called *agents*. What is an agent? Despite numerous attempts to answer this question, there seems to be no conclusive definition. We assume that MAS are, most of all, a philosophical metaphor that induces a specific way of seeing the world, and makes us use agent-oriented vocabulary when describing the phenomena we are interested in. Thus, while some researchers present multi-agent systems as a new paradigm for computation or design, we believe that primarily multi-agent systems form a new paradigm for *thinking* and *talking* about the world, and assigning it a specific conceptual structure. The metaphor of a multi-agent system seems to build on the intuition that *we* are agents – and that other entities we study can be just like us to some extent. The usual properties of agents, like autonomy, pro-activeness etc., seem to be secondary: they are results of an introspection rather than primary assumptions we start with.

We note that this view of multi-agent systems comes close to the role of *logic* in both philosophy and computer science. Logic provides conceptual structures for *modeling* and *reasoning* about the world in a precise manner – and, occasionally, it also provides tools to do it automatically.

References: Reader interested in issues related to multi-agent systems is referred to [30, 29].

1.2 Modal Logic

Modal logic is an extension of classical logic with new operators $\Box$ (*necessity*) and $\Diamond$ (*possibility*): $\Box p$ means that p is true in every possible scenario, while $\Diamond p$ means that p is true in at least one possible scenario. Let $\mathcal{PV}$ be the set of *propositional variables* (also called *propositions*). Models of modal logics are called *Kripke models* or *possible world models*, and include the set of *possible worlds* (or states) St, modal accessibility relation $\mathcal{R} \subseteq St \times St$, and interpretation of the

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

propositions $V : \mathcal{PV} \to 2^{St}$. Now, for a model $M = (St, \mathcal{R}, V)$ and world q in M:

$$M, q \models \Box\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q\mathcal{R}q'$$
$$M, q \models \Diamond\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for some } q' \text{ such that } q\mathcal{R}_i q'$$

Modal logic can be further extended to multi-modal logic, where we deal with several modal operators: $\Box_i$ and $\Diamond_i$, each of them interpreted over the corresponding i-modal accessibility relation $\mathcal{R}_i \subseteq St \times St$.

The actual accessibility relations can capture various dimensions of the reality (and therefore give rise to different kinds of modal logics): knowledge ($\rightsquigarrow$ *epistemic logic*), beliefs ($\rightsquigarrow$ *doxastic logic*), obligations ($\rightsquigarrow$ *deontic logic*), actions ($\rightsquigarrow$ *dynamic logic*), time ($\rightsquigarrow$ *temporal logic*) etc. In particular, various aspects of agents (and agent systems) can be naturally captured within this generic framework.

References: A gentle introduction to modal logic can be found in [1].

1.3 Reasoning about Knowledge

The basic *epistemic logic* which we consider here involves modalities for individual agent's knowledge K_i, with $\mathrm{K}_i\varphi$ interpreted as "agent i knows that φ". Additionally, one can consider modalities for collective knowledge of groups of agents: *mutual knowledge* ($\mathrm{E}_A\varphi$: "everybody in group A knows that φ"), *common knowledge* ($\mathrm{C}_A\varphi$: "all the agents in A know that φ, and they know that they know it etc."), and *distributed knowledge* ($\mathrm{D}_A\varphi$: "if the agents could share their individual information, they would be able to recognize that φ"). Note that $\mathrm{E}_A\varphi \equiv \bigwedge_{i\in A} \mathrm{K}_i\varphi$, and hence the operators for mutual knowledge can be omitted from the language.

The formal semantics for the logic is based on *multi-agent epistemic models* of the type $\langle St, \sim_1, ..., \sim_k, V\rangle$, where St is a set of *epistemic states*, V is a valuation of propositions, and each $\sim_i \subseteq St \times St$ is an equivalence relation that defines the *indistinguishability of states* for agent i. Operators K_i are provided with the usual Kripke semantics given by the clause:

$$M, q \models \mathrm{K}_i\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_i q'$$

The accessibility relation corresponding to E_A is defined as $\sim_A^{\mathrm{E}} = \bigcup_{i\in A} \sim_i$, and the semantics of E_A becomes

$$M, q \models \mathrm{E}_A\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_A^{\mathrm{E}} q'.$$

Common knowledge C_A is given semantics in terms of the relation $\sim_A^{\mathrm{C}}$ defined as the transitive closure of $\sim_A^{\mathrm{E}}$:

$$M, q \models \mathrm{C}_A\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_A^{\mathrm{C}} q'.$$

Finally, distributed knowledge $\mathrm{D_A}$ is given semantics in terms of the relation $\sim_A^{\mathrm{D}}$ defined as $\bigcap_{i\in A} \sim_i$, following the same pattern:

$$M, q \models \mathrm{D}_A\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_A^{\mathrm{D}} q'.$$

Example 1.1 (Robots and Carriage)**.** *Consider the scenario depicted in Figure 1.1. Two robots push a carriage from opposite sides. As a result, the carriage can move clockwise or anticlockwise, or it can remain in the same place – depending on who pushes with more force (and, perhaps, who refrains from pushing). To make our model of the domain discrete, we identify 3 different*

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

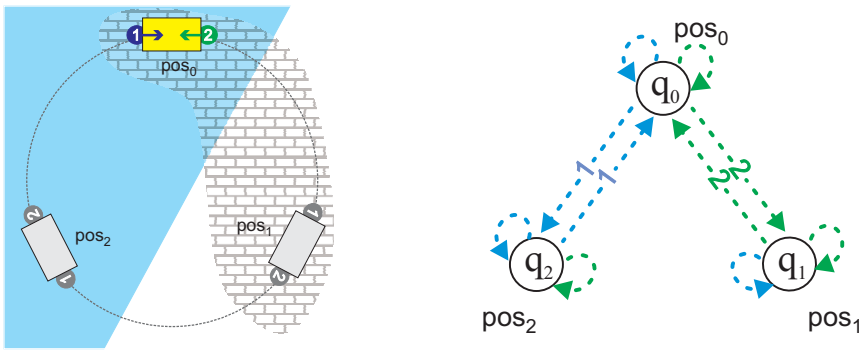

Figure 1.1: Two robots and a carriage: a schematic view (left) and a Kripke model M_1 of the robots' knowledge (right).

positions of the carriage, and associate them with states q_0, q_1, *and* q_2. *We label the states with propositions* $\mathsf{pos}_0, \mathsf{pos}_1, \mathsf{pos}_2$, *respectively, to allow for referring to the current position of the carriage in the object language.*

Moreover, we assume that robot 1 *is only able to observe the color of the surface on which it is standing, and robot* 2 *perceives only the texture. As a consequence, the first robot can distinguish between position* 0 *and position* 1, *but positions* 0 *and* 2 *look the same to it. Likewise, the second robot can distinguish between positions* 0 *and* 2, *but not* 0 *and* 1. *In the resulting epistemic model, we have for instance that* $M_1, q_0 \models \neg \mathrm{K}_1 \mathsf{pos}_0 \wedge \neg \mathrm{K}_1 \mathsf{pos}_2 \wedge \mathrm{K}_1(\mathsf{pos}_0 \vee \mathsf{pos}_2)$*: the first robot knows that the position is either* 0 *or* 2, *but not which of them precisely. Moreover,* $M_1, q_0 \models \mathrm{K}_1 \neg \mathsf{pos}_1$ *(robot* 1 *knows that the current position is not* 1*). The robot also knows that the other agent can distinguish between smooth and rough texture:* $M_1, q_0 \models \mathrm{K}_1\big((\mathsf{pos}_2 \to \mathrm{K}_2 \mathsf{pos}_2) \wedge (\neg \mathsf{pos}_2 \to \mathrm{K}_2 \neg \mathsf{pos}_2)\big)$. *Finally, if the robots share their information, they know the current position precisely:* $M_1, q_0 \models \mathrm{D}_{\{1,2\}} \mathsf{pos}_0$.

Note that epistemic models are usually constructed from the point of view of an *external omniscient observer*, most typically a system designer who has a complete view of the entire system.

References: [9] presents what has become the standard treatment of reasoning about knowledge within the computer science community. More lightweight surveys can be found in [11, 27].

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Chapter 2

Reasoning about Evolution of Systems

2.1 Modal Logics of Time and Action

We present a brief overview of logics that regard the dynamics of systems. That is, essentially, logics that focus on actions that can be performed by (or in) a system, and on the way in which the system can evolve over time. We consider two basic cases here: either actions are first-class citizens of the language, or we abstract from them and reason about *change* in general.

PDL. *Propositional Dynamic Logic (PDL)*, which was primarily designed to reason about computer programs, is probably the most typical representative of logics with explicit actions. Actions are represented in the language with *action labels* $\alpha_1, \alpha_2, ...$ from a finite set *Act*. Complex action terms can be also constructed, for example sequential composition $(\alpha_1; \alpha_2)$, nondeterministic choice $(\alpha_1 \cup \alpha_2)$, finite iteration $(\alpha*)$ etc. Now, $[\alpha]\varphi$ expresses the fact that φ is bound to hold after *every* execution of α, and $\langle\alpha\rangle\varphi \equiv \neg[\alpha]\neg\varphi$ says that φ holds after *at least one* possible execution of α. On the semantic side, we have *Labeled Transition Systems* $M = \langle St, \stackrel{\alpha_1}{\rightarrow}, \stackrel{\alpha_2}{\rightarrow}, ..., V\rangle$, where actions are modeled as (nondeterministic) state transformations $\stackrel{\alpha}{\rightarrow} \subseteq St \times St$, and the following semantic clause:

$$M, q \models [\alpha]\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q \stackrel{\alpha}{\rightarrow} q'.$$

We mention PDL only in passing here, as it will not be used in the rest of the materials.

LTL. *Temporal logics* leave actions implicit, and instead focus on possible patterns of evolution. Typical temporal operators are: X ("in the next state"), G ("always from now on"), F ("sometime in the future"), and U ("until"). Models include one transition relation, and come in two versions. *Linear time* models define a total ordering on possible worlds ("time moments"), so that a model can be seen as a single infinite path λ with successive states $\lambda[0], \lambda[1], ...$; by λ_i we denote the suffix of λ starting from the state $\lambda[i]$. The semantics of *Linear Temporal Logic (LTL)* can be

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

defined as follows:

$$\begin{aligned}
\lambda \models p &\quad \text{iff} \quad \lambda[0] \in V(p);\\
\lambda \models \neg\varphi &\quad \text{iff} \quad \lambda \not\models \varphi,\\
\lambda \models \varphi \wedge \psi &\quad \text{iff} \quad \lambda \models \varphi \text{ and } \lambda \models \psi;\\
\lambda \models \mathrm{X}\varphi &\quad \text{iff} \quad \lambda_1 \models \varphi;\\
\lambda \models \varphi \mathrm{U} \psi &\quad \text{iff} \quad (\exists j \geq 0)\big(\lambda_j \models \psi \text{ and } (\forall 0 \leq i < j)\ \lambda_i \models \varphi\big).
\end{aligned}$$

with $\mathrm{F}\varphi \equiv \mathbf{true}\mathrm{U}\varphi$ and $\mathrm{G}\varphi \equiv \neg\mathrm{F}\neg\varphi$, where $\mathbf{true} \stackrel{def}{=} p \vee \neg p$, for some $p \in \mathcal{PV}$.

Example 2.1. *Consider the path* $(q_0q_1q_2)^\omega = q_0q_1q_2q_0q_1q_2q_0q_1q_2\ldots$ *from the robots and carriage scenario (Example 1.1). For that path, we have for instance that* $\mathrm{F}\mathsf{pos}_2$ *(position 2 will be eventually achieved), and even* $\mathrm{GF}\mathsf{pos}_2$ *(position 2 will be achieved infinitely many times). However, it is not the case that the carriage will stop and stay in position 0 for good:* $\neg\mathrm{FG}\mathsf{pos}_2$.

Typically, when LTL is used for specifying properties of a system, the formulae are interpreted over all the infinite paths from a transition model of the system.

CTL*. *Branching-time models*, on the other hand, consist of a tree that encapsulates all possible evolutions of the system. *Computation Tree Logic (CTL*)* extends LTL with *path quantifiers* E ("there is a path"), and A ("for every path"). Formally, the syntax and semantics of CTL* is defined in the following way. Let $\mathcal{PV} = \{p_1, p_2\ldots\}$ be a set of propositional variables. The language of CTL* is given as the set of all the state formulas φ (interpreted at states of a model), defined using path formulas γ (interpreted at paths of a model), by the following grammar:

$$\begin{aligned}
\varphi &:= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{A}\gamma \mid \mathsf{E}\gamma\\
\gamma &:= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \mathrm{X}\gamma \mid \gamma\mathrm{U}\gamma.
\end{aligned}$$

In the above $p \in \mathcal{PV}$, A ('for All paths') and E ('there Exists a path') are path quantifiers, whereas X ('neXt') and U ('Until') are temporal operators like before.

Obviously, when verifying properties of a particular system, a model given in the form of a set of infinite paths (for LTL) or an infinite tree (CTL*) would be impractical. Instead, the behavior of the system is usually represented as a Kripke model of transitions – either labeled (with multiple transition relations, one per action name) or unlabeled (all transitions "collapsed" into a single relation $\rightarrow$). Therefore, semantics of CTL* is frequently defined in terms of standard Kripke models. Notice that the unfolding of a Kripke model is an inifinite tree.

Let $M = (St, \rightarrow, V)$ be a Kripke model. For $q_0 \in St$ a (full) *path* $\lambda = (q_0, q_1, \ldots)$ is an infinite sequence of states in St starting at q_0, where $q_i \rightarrow q_{i+1}$ for all $i \geq 0$, and $\lambda_i = (q_i, q_{i+1}, \ldots)$ is the i-th suffix of λ starting at q_i. By $M, q \models \varphi$ ($M, \lambda \models \gamma$) we mean that φ holds in the state q (γ holds on the path λ, respectively) of the model M. In what follows the model M is sometimes omitted if it is clear from the context. The relation $\models$ is defined inductively below.

$$\begin{aligned}
M, q \models p &\quad \text{iff} \quad q \in V(p), \text{ for } p \in \mathcal{PV},\\
M, x \models \neg y &\quad \text{iff} \quad M, x \not\models y, \text{ for } x \in \{q, \lambda\}, y \in \{\varphi, \gamma\},\\
M, x \models y_1 \wedge y_2 &\quad \text{iff} \quad M, x \models y_1 \text{ and } M, x \models y_2, \text{ for } x, y \text{ as above},\\
M, q \models \mathrm{A}\gamma &\quad \text{iff} \quad M, \lambda \models \gamma \text{ for each path } \lambda \text{ starting at } q,\\
M, q \models \mathrm{E}\gamma &\quad \text{iff} \quad M, \lambda \models \gamma \text{ for some path } \lambda \text{ starting at } q,\\
M, \lambda \models \varphi &\quad \text{iff} \quad M, \lambda[0] \models \varphi, \text{ for a state formula } \varphi,\\
M, \lambda \models \mathrm{X}\gamma &\quad \text{iff} \quad M, \lambda_1 \models \gamma,\\
M, \lambda \models \gamma_1\mathrm{U}\gamma_2 &\quad \text{iff} \quad (\exists j \geq 0)\big(M, \lambda_j \models \gamma_2 \text{ and } (\forall 0 \leq i < j)\ M, \lambda_i \models \gamma_1\big).
\end{aligned}$$

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

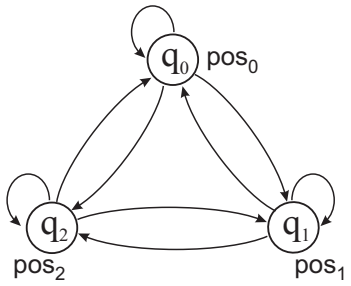

Figure 2.1: Two robots and a carriage: transition system M_2 that models possible movements of the carriage.

The logic CTL is an important subset of $\mathrm{CTL}^*_{-\mathrm{X}}$. In "pure" (or "vanilla") CTL every occurrence of a path quantifier is followed by exactly one temporal operator, so there are only state formulae. Note that in the case of "vanilla" CTL, there exists an alternative semantics given entirely in terms of satisfaction of formulae in *states*:

$M, q \models p$ iff $q \in V(p)$;

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$;

$M, q \models \mathsf{EX}\varphi$ iff $M, \lambda[1] \models \varphi$ for some path λ starting from q;

$M, q \models \mathsf{EG}\varphi$ iff there is a path λ starting from q such that $(\forall i \geq 0)\ M, \lambda[i] \models \varphi$;

$M, q \models \mathsf{E}\varphi\mathrm{U}\psi$ iff there is a path λ starting from q such that $(\exists j \geq 0)\big(M, \lambda[j] \models \psi$ and $(\forall 0 \leq i < j)\ M, \lambda[i] \models \varphi\big)$.

Example 2.2. *A possible transition system for the robots scenario is depicted in Figure 2.1. In that model, we have for instance* $M_2, q_0 \models \mathsf{EF}\,\mathsf{pos}_1$*: in state* q_0*, there is a path such that the carriage will reach position* 1 *sometime in the future. The same is clearly not true for* all *paths, so we also have that* $M_2, q_0 \models \neg\mathsf{AF}\,\mathsf{pos}_1$.

Temporal and dynamic dimensions have been combined with other modalities, e.g. in the well-known *BDI logics* of beliefs, desires, and intentions [5, 22]. We will present simple extensions of LTL and CTL with epistemic modalities in Section 2.2, and discuss verification of temporal-epistemic logic in later sections.

Modal μ-calculus. Propositional modal μ-calculus was introduced by D. Kozen [15]. Let $\mathcal{PV}$ be a set of propositional variables and FV be a set of fixed-point variables. The language of modal μ-calculus L_μ is defined by the following grammar:

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{AX}\varphi \mid \mathsf{EX}\varphi \mid Z \mid \mu Z.\varphi(Z) \mid \nu Z.\varphi(Z),$$

where p ranges over $\mathcal{PV}$, Z – over FV, and $\varphi(Z)$ is a modal μ-calculus formula syntactically monotone in the fixed-point variable Z, i.e., all the free occurrences of Z in $\varphi(Z)$ fall under an even number of negations.

Let $M = (St, \rightarrow, V)$ be a Kripke model. Notice that the set 2^{St} of all subsets of St forms a lattice under the set inclusion ordering. Each element St' of the lattice can also be thought

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego
KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

of as a *predicate* on St, where this predicate is viewed as being true for exactly the states in St'. The least element in the lattice is the empty set, which we also refer to as **false**, and the greatest element in the lattice is the set St, which we sometimes write as **true**. A function ζ mapping 2^{St} to 2^{St} is called a *predicate transformer*. A set $St' \subseteq St$ is a *fixed point* of a function $\zeta : 2^{St} \longrightarrow 2^{St}$ if

$$\zeta(St') = St'.$$

Whenever ζ is *monotonic*, i.e., $S_1 \subseteq S_2$ implies $\zeta(S_1) \subseteq \zeta(S_2)$, where $S_1, S_2 \subseteq St$, it has the least fixed point denoted $\mu Z.\zeta(Z)$ and the greatest fixed pointdenoted $\nu Z.\zeta(Z)$. When $\zeta(Z)$ is also $\bigcup$-continuous, i.e., $S_1 \subseteq S_2 \subseteq \ldots$ implies $\zeta(\bigcup_{i\geq 0} S_i) = \bigcup_{i\geq 0} \zeta(S_i)$, then

$$\mu Z.\zeta(Z) = \bigcup_{i\geq 0} \zeta^i(\mathbf{false}).$$

When $\zeta(Z)$ is also $\bigcap$-continuous, i.e., $S_1 \supseteq S_2 \supseteq \ldots$ implies $\zeta(\bigcap_{i\geq 0} S_i) = \bigcap_{i\geq 0} \zeta(S_i)$, then

$$\nu Z.\zeta(Z) = \bigcap_{i\geq 0} \zeta^i(\mathbf{true})$$

(see [26]).

The semantics of L_μ is given inductively for each formula φ of L_μ, a model M, a valuation $\mathcal{E} : FV \longrightarrow 2^S$ of the fixed-point variables (called an *environment*), and a state $q \in St$:

$$\begin{array}{lll}
M, \mathcal{E}, q \models p & \text{iff} & q \in V(p), \text{ for } p \in \mathcal{PV}, \\
M, \mathcal{E}, q \models \neg\varphi & \text{iff} & M, \mathcal{E}, q \not\models \varphi, \\
M, \mathcal{E}, q \models \varphi \wedge \psi & \text{iff} & M, \mathcal{E}, q \models \varphi \text{ and } M, \mathcal{E}, q \models \psi, \\
M, \mathcal{E}, q \models \mathrm{AX}\varphi & \text{iff} & (\forall q' \in St)((q \rightarrow q') \Rightarrow (M, \mathcal{E}, q' \models \varphi)), \\
M, \mathcal{E}, q \models \mathrm{EX}\varphi & \text{iff} & (\exists q' \in St)((q \rightarrow q') \wedge M, \mathcal{E}, q' \models \varphi), \\
M, \mathcal{E}, q \models Z & \text{iff} & q \in \mathcal{E}(Z), \text{ for } Z \in FV, \\
M, \mathcal{E}, q \models \mu Z.\varphi(Z) & \text{iff} & q \in \bigcap\{U \subseteq St \mid \\
 & & \quad \{q' \in St \mid M, \mathcal{E}[Z \leftarrow U], q' \models \varphi\} \subseteq U\}, \\
M, \mathcal{E}, q \models \nu Z.\varphi(Z) & \text{iff} & q \in \bigcup\{U \subseteq St \mid \\
 & & \quad U \subseteq \{q' \in St \mid M, \mathcal{E}[Z \leftarrow U], q' \models \varphi\}\},
\end{array}$$

where $\varphi, \psi \in L_\mu$, and $\mathcal{E}[Z \leftarrow U]$ is like $\mathcal{E}$ except that it maps Z to U. Now, we define $M, q \models \varphi$ iff $M, \mathcal{E}, q \models \varphi$ for each environment $\mathcal{E}$.

It is known that both CTL and CTL* can be translated into modal μ-calculus [15]. For example, we give characterisations of basic CTL modalities in terms of modal μ-calculus formulas:

- $\mathrm{A}(\varphi \mathrm{U} \psi) \equiv \mu Z.(\psi \vee (\varphi \wedge \mathrm{AX}Z))$,
- $\mathrm{E}(\varphi \mathrm{U} \psi) \equiv \mu Z.(\psi \vee (\varphi \wedge \mathrm{EX}Z))$,
- $\mathrm{AG}\varphi \equiv \nu Z.(\varphi \wedge \mathrm{AX}Z)$,
- $\mathrm{EG}\varphi \equiv \nu Z.(\varphi \wedge \mathrm{EX}Z)$.

The translation of CTL* to modal μ-calculus is more involved and can be found in [7]. It is worth noticing that the translations are important in practice because correctness specifications written in logics such as CTL or CTL* are often much more readable than specifications written directly in modal μ-calculus.

References: Dynamic logic is treated extensively in [12]; readers interested in temporal logic are referred to [6, 10]. An introduction to μ-calculus can be found in [25].

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

2.2 Combining Time and Knowledge

In this section we discuss how the interaction between knowledge and time can be captured in modal logic. To this end, we combine epistemic and temporal dimensions in a multi-modal logic. The main proposals that we are going to consider is LTLK and CTLK, which are straightforward combinations of respectively LTLK, CTL, and standard epistemic logic.

LTLK. The language of LTLK includes all the operators of LTL and epistemic logic. Moreover, its semantics is based on Kripke models which include both temporal and epistemic accessibility relations: $M = \langle St, R, \sim_1, ..., \sim_k, V\rangle$. The semantics of LTLK simply combines the clauses for LTL and modal epistemic logic:

$$\begin{aligned}
\lambda \models p \quad &\text{iff} \quad \lambda[0] \in V(p);\\
\lambda \models \neg\varphi \quad &\text{iff} \quad \lambda \not\models \varphi,\\
\lambda \models \varphi \wedge \psi \quad &\text{iff} \quad \lambda \models \varphi \text{ and } \lambda \models \psi;\\
\lambda \models \mathrm{X}\varphi \quad &\text{iff} \quad \lambda_1 \models \varphi;\\
\lambda \models \varphi \mathrm{U} \psi \quad &\text{iff} \quad (\exists j \geq 0)\big(\lambda_j \models \psi \text{ and } (\forall 0 \leq i < j)\ \lambda_i \models \varphi\big);\\
\lambda \models \mathrm{K}_i\varphi \quad &\text{iff} \quad \lambda'_j \models \varphi \text{ for all paths } \lambda' \text{ and all } (j \geq 0) \text{ s.t. } \lambda[0] \sim_i \lambda'[j],
\end{aligned}$$

where $1 \leq i \leq k$ and $M, q \models \varphi$ iff $M, \lambda_0 \models \varphi$ for all the paths λ starting at q.

CTLK. The language of CTLK includes all the operators of CTL and epistemic logic. Moreover, its semantics is based on Kripke models which include both temporal and epistemic accessibility relations: $M = \langle St, R, \sim_1, ..., \sim_k, V\rangle$. Like for LTLK, the semantics of CTLK takes the union of both respective semantics:

$M, q \models p$ iff $q \in V(p)$;

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$;

$M, q \models \mathsf{EX}\varphi$ iff $M, \lambda[1] \models \varphi$ for some path λ starting from q;

$M, q \models \mathsf{EG}\varphi$ iff there is a path λ starting from q such that $(\forall i \geq 0)\ M, \lambda[i] \models \varphi$;

$M, q \models \mathsf{E}\varphi\mathrm{U}\psi$ iff there is a path λ starting from q such that $(\exists j \geq 0)\big(M, \lambda[j] \models \psi$ and $(\forall 0 \leq i < j)\ M, \lambda[i] \models \varphi\big)$.

$M, q \models \mathrm{K}_i\varphi$ iff $M, q \models \varphi$ for every q' such that $q \sim_i q'$, for $1 \leq i \leq k$.

Example 2.3. *Let M_3 be the temporal-epistemic model of robots and carriage that combines states and relations from models M_1, M_2 (Figures 1.1 and 2.1). Now, $M_3, q_0 \models \mathrm{K}_1\mathsf{EF}\,\mathsf{pos}_2$: robot 1 knows that the carriage can get to position 2 eventually. Moreover, $M_3, q_0 \models \mathsf{EX}\bigvee_{i=0,1,2}(\mathsf{pos}_i \rightarrow \mathrm{K}_1\mathsf{pos}_i)$: it is possible that in the next moment robot 1 will know its position precisely.*

Interpreted systems. A well known study of the interplay between knowledge and time has been presented in [9]. The proposal builds on a notion of *global states*, defined formally as follows. Firstly, each agent $i \in \mathbb{A}\mathrm{gt} = \{1, ..., k\}$ has a set of *local states* St_i. Every *global state* is represented by a tuple of local states $\langle q_1, ..., q_k\rangle$ corresponding to all agents. Thus, the global state space St is (a subset of) the product $St_1 \times ... \times St_k$. It has been argued in many places that interpreted systems provide more "grounded" semantics for agents' knowledge than abstract

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Kripke models. This is because starting from the local state spaces makes it clearer how the epistemic model should be actually constructed.

It is usually assumed in interpreted systems that each agent has access only to its own local state, i.e.:

$$\langle q_1, ..., q_k \rangle \sim_i \langle q'_1, ..., q'_k \rangle \text{ iff } q_i = q'_i.$$

The temporal dimension is added by considering *runs*, i.e., sequences of global states:

$$r : \mathbb{N} \to St_1 \times ... \times St_k.$$

A *system* is a set of such runs. An *interpreted system* is a system plus a valuation of propositions: $\langle \mathcal{R}, V \rangle$. Given an interpreted system $\mathcal{I} = \langle \mathcal{R}, V \rangle$, a *point* in $\mathcal{I}$ is a pair $\langle r, m \rangle$ where r is a run and $m \in \mathbb{N}$. Epistemic equivalence between points is defined as follows:

$$\langle r, m \rangle \sim_i \langle r', m' \rangle \text{ iff } r(m) \sim_i r'(m').$$

Now, all epistemic modalities can be interpreted as before, e.g.:

$$\mathcal{I}, r, m \models \mathrm{K}_i\varphi \text{ iff } \mathcal{I}, r', m' \models \varphi \text{ for all } \langle r', m' \rangle \text{ such that } \langle r, m \rangle \sim_i \langle r', m' \rangle.$$

Moreover, the standard temporal operators of LTL can be interpreted, too:

- $\mathcal{I}, r, m \models \mathrm{X}\varphi$ iff $\mathcal{I}, r, m+1 \models \varphi$,
- $\mathcal{I}, r, m \models \varphi \mathrm{U} \psi$ iff $\mathcal{I}, r, m' \models \psi$ for some $m' > m$ and $\mathcal{I}, r, m'' \models \varphi$ for all m'' such that $m \leq m'' < m'$.

Finally, the semantics of path quantifiers from CTL* can be defined in interpreted systems as follows:

- $\mathcal{I}, r, m \models \mathsf{E}\varphi$ iff there is r' such that $r'[0..m] = r[0..m]$ and $\mathcal{I}, r', m \models \varphi$.

It should be pointed out that the semantics of knowledge presented above is only one of several possibilities. It encodes the assumption that agents are *memoryless* in the sense that they do not have "external" memory of past events; the whole memory of an agent is encapsulated in its local state. The other extreme is to assume that local states represent the agents' observations rather than knowledge, and that agents have *perfect recall* of everything that has been observed. If we additionally assume the existence of a global universally accessible clock, the semantics of knowledge can be defined as follows:

$$\langle r, m \rangle \approx_i \langle r', m' \rangle \text{ iff } m = m' \text{ and } r(j) \sim_i r'(j) \text{ for all } j \leq m,$$
$$\mathcal{I}, r, m \models \mathrm{K}_i\varphi \text{ iff } \mathcal{I}, r', m' \models \varphi \text{ for all } \langle r', m' \rangle \text{ such that } \langle r, m \rangle \approx_i \langle r', m' \rangle.$$

Interpreted systems have been applied to modeling of *distributed systems, knowledge bases, message passing systems*, and more specifically to phenomena like *perfect recall, synchrony and asynchrony* etc.

References: Possible interactions between the temporal and epistemic dimensions are studied extensively in [9].

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Chapter 3

Introduction to Model Checking for Knowledge and Time

In this section, we look at standard non-symbolic model checking algorithms for temporal and temporal-epistemic logics. We start with some notes on verification and complexity classes in Section 3.1. Basic algorithms and complexity results for verification of temporal logics are presented in Section 3.2. Then, we discuss the model checking algorithm exploiting the fixed-point characterization of CTLK in Section 3.3. Finally we briefly present the approach of *symbolic model checking* based on a translation to Ordered Binary Decision Diagrams.

3.1 Complexity of Verificatin

Within the materials we consider both practical algorithms for verifying properties of MAS, and the theoretical complexity of these problems. Here is a short (and rather informal) description of the most relevant complexity classes. A formal introduction can be found in any textbook on complexity theory (cf. e.g. [19]).

- **P**: problems solvable in *polynomial time* by a *deterministic* Turing machine,
- **NP**: problems solvable in *polynomial time* by a *nondeterministic* Turing machine,
- $\mathbf{\Sigma_n^P}$ / $\mathbf{\Pi_n^P}$ / $\mathbf{\Delta_n^P}$: problems solvable in polynomial time with use of adaptive queries to an *n-level oracle*,
- **PSPACE**: problems solvable by queries to a multilevel oracle with unbounded "height",
- **EXPTIME**: problems solvable in *exponential time* by a deterministic Turing machine.

Of course, theoretical complexity has many deficiencies: it refers only to the worst (hardest) instance in the set, neglects coefficients in the function characterizing the complexity, etc. However, it often gives a good indication of the inherent hardness of the problem in terms of *scalability*. For low complexity classes, scaling up from small instances of the problem to larger instances is relatively easy. For high complexity classes, this is not the case anymore.

The process of verification (so called *model checking*) answers whether a given formula φ is satisfied in a state q of model M. Formally, *local model checking* is the decision problem that determines membership in the set

$$\mathsf{MC}(\mathcal{L}, \mathsf{Struc}, \models) = \{(M, q, \varphi) \in \mathsf{Struc} \times St \times \mathcal{L} \mid M, q \models \varphi\},$$

where $\mathcal{L}$ is a logical language, Struc is a class of (pointed) models for $\mathcal{L}$, and $\models$ is a semantic satisfaction relation compatible with $\mathcal{L}$ and Struc.[1]

It is often useful to compute the set of states in M that satisfy formula φ instead of checking if φ holds in a particular state. This variant of the problem is known as *global model checking*. It is easy to see that, for the settings we consider here, the complexities of local and global model checking coincide, and the algorithms for one variant of model checking can be adapted to the other variant in a simple way. As a consequence, we will use both notions of model checking interchangeably.[2]

References: For a comprehensive textbook on model checking, see e.g. [4].

3.2 Verifying Temporal Logic

An excellent survey on the model checking complexity of temporal logics has been presented in [23]. Here, we only recall the most relevant results before turning our focus to actual algorithms in Section 3.

Let M be a Kripke model and q be a state in the model. Model checking a CTL formula φ in M, q determines whether $M, q \models \varphi$, i.e., whether φ holds in M, q. The same applies to model checking CTL*. For LTL, checking $M, q \models \varphi$ means that we check the *validity* of φ in the pointed model M, q, i.e., whether φ holds *on all the paths* in M that start from q (equivalent to CTL* model checking of formula $\mathsf{A}\varphi$ in M, q.

It has been known for a long time that the formulae of CTL can be model-checked in time linear with respect to the size of the model and the length of the formula [3], whereas formulae of LTL and CTL* are significantly harder to verify. The size of the model M, denoted by $|M|$, is defined by the sum of the number of its states and its transitions $|St| + |R|$.

Theorem 3.1 (CTL [3, 23])**.** *Model checking CTL is* **P***-complete, and can be done in time* $\mathbf{O}(|M| \cdot |\varphi|)$.

Sketch. The algorithm (for an extension of CTL) determining the states in a model at which a given formula holds is presented in Section 3.3. The lower bound (**P**-hardness) can be for instance proven by a reduction of the tiling problem [23]. □ □

Theorem 3.2 (LTL [24, 16, 28])**.** *Model checking LTL is* **PSPACE***-complete, and can be done in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|M|)$.

Sketch. We sketch here the approach of [28]. Firstly, given an LTL-formula φ, a Büchi automaton $\mathcal{A}_{\neg\varphi}$ of size $2^{\mathbf{O}(|\varphi|)}$ accepting exactly the runs (infinite paths of states) satisfying $\neg\varphi$ is constructed. The pointed Kripke model M, q can directly be interpreted as a Büchi automaton $\mathcal{A}_{M,q}$ of size $\mathbf{O}(|M|)$ accepting all possible runs in the Kripke model starting in q. Then, the model checking problem reduces to the non-emptiness check of $L(\mathcal{A}_{M,q}) \cap L(\mathcal{A}_{\neg\varphi})$, which can be done in time $\mathbf{O}(|M|) \cdot 2^{\mathbf{O}(|\varphi|)}$ by constructing the product automaton. Notice that the non-emptiness can be checked in linear time w.r.t. to the size of the automaton. A **PSPACE**-hardness proof can be for instance found in [24]. □ □

[1] We omit parameters if they are clear from the context.

[2] The only logic mentioned in the materials, for which the complexities of global and local model checking differ, is Constructive Strategic Logic from Section **??**. However, we do not discuss the model checking problem for CSL in the materials.

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

MC	m, l
Epistemic logic	**P**-complete
PDL	**P**-complete
CTL	**P**-complete
LTL	**PSPACE**-complete
CTL*	**PSPACE**-complete

SAT	l
Epistemic logic	**PSPACE**-complete
PDL	**PSPACE**-complete
CTL	**EXPTIME**-complete
LTL	**PSPACE**-complete
CTL*	**2EXPTIME**-complete

Figure 3.1: Basic complexity results: model checking (left) and satisfiability (right)

The hardness of CTL* model checking is immediate from Theorem 3.2 as LTL can be seen as a fragment of CTL*. For the proof of the upper bound one combines the CTL and LTL model checking techniques. Consider a CTL* formula φ which contains a state subformula $\mathsf{E}\psi$, where ψ is a pure LTL formula. Firstly, we can use the LTL model checking to determine all the states which satisfy $\mathsf{E}\psi$ (these are all states q in which the LTL formula $\neg\psi$ is *not* true) and label them by a fresh propositional symbol, say p, and replace $\mathsf{E}\psi$ in φ by p as well. Applying this procedure recursively yields a pure CTL formula, which can be verified in polynomial time. Hence, the procedure can be implemented by an oracle machine of type $\mathbf{P}^{\mathbf{PSPACE}} = \mathbf{PSPACE}$ (the LTL model checking algorithm might be employed polynomially many times). Thus, the complexity for CTL* is the same as for LTL.

Theorem 3.3 (CTL* [3, 8]). *Model checking CTL* is* **PSPACE***-complete, and can be done in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|M|)$.

Figure 3.1 presents the basic complexity results for model checking of temporal and epistemic logics. For comparison, we also list complexities of analogous satisfiability problems. The input of the SAT problem is given as a formula of the logic, and its size is measured in the length of the formula. The input of the model checking problem is given as the formula and the model; the size of an input instance is measured as $m \cdot l$, where m is the number of transitions in the model, and l is the length of the formula.

References: Readers interested in basic complexity results for model checking temporal logics are referred to the excellent survey [23].

3.3 Fixed-point Verification for CTL and CTLK

Symbolic and non-symbolic model checking methods can exploit the fixed-point characterization of CTLK formulas. These methods operate on sets of states contrary to the state labeling algorithm operating on single states of the model. In what follows by $\overline{\mathrm{C}}_\mathrm{A}$ we denote the modality dual to C_A, i.e., $\overline{\mathrm{C}}_\mathrm{A}\varphi \stackrel{def}{=} \neg\mathrm{C}_\mathrm{A}\neg\varphi$. Similarly, for $\overline{\mathrm{E}}_\mathrm{A}$ and $\overline{\mathrm{D}}_\mathrm{A}$. We do not discuss here algorithms for the operators $\overline{\mathrm{E}}_\mathrm{A}$ and $\overline{\mathrm{D}}_\mathrm{A}$ as these are straightforward given an algorithm for $\overline{\mathrm{K}}_i$. Then, labelling of the states with the subformulas or computation of OBDD representation of a CTLK formula uses the standard algorithms for computing the minimal and the maximal fixpoints as follows.

- $\mathrm{EG}\varphi \equiv \varphi \wedge \mathrm{EXEG}\varphi$,
- $\mathrm{E}(\varphi\mathrm{U}\psi) \equiv \psi \vee (\varphi \wedge \mathrm{E}(\varphi\mathrm{U}\psi))$,
- $\overline{\mathrm{C}}_\mathrm{A}\varphi \equiv \varphi \vee \overline{\mathrm{E}}_\mathrm{A}\overline{\mathrm{C}}_\mathrm{A}\varphi$.

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego
KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Let $[\![\varphi]\!] = \{q \in St \mid q \models \varphi\}$. Then, we have:

- $[\![EG\varphi]\!] = [\![\varphi]\!] \cap [\![\mathrm{EXEG}\varphi]\!]$,
- $[\![\mathrm{E}(\varphi \mathrm{U} \psi)]\!] = [\![\psi]\!] \cup ([\![\varphi]\!] \cap [\![\mathrm{EXE}(\varphi \mathrm{U} \psi)]\!])$
- $[\![\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!] = [\![\varphi]\!] \cup [\![\overline{\mathrm{E}}_{\mathrm{A}}\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!])$

For each subset $X \subseteq St$, we can easily define algorithms computing the following sets:

- $pre(X) = \{q \in St \mid (\exists q' \in X)\ q \to q'\}$,
- $indis_i(X) = \{q \in St \mid (\exists q' \in X)\ q \sim_i q'\}$, and
- $indis^E_{\mathrm{A}}(X) = \{q \in St \mid (\exists q' \in X)\ q \sim^E_{\mathrm{A}} q'\}$.

Then, we have:

- $[\![EG\varphi]\!] = [\![\varphi]\!] \cap pre([\![\mathrm{EG}\varphi]\!])$,
- $[\![\mathrm{E}(\varphi \mathrm{U} \psi)]\!] = [\![\psi]\!] \cup ([\![\varphi]\!] \cap pre([\![\mathrm{E}(\varphi \mathrm{U} \psi)]\!]))$,
- $[\![\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!] = [\![\varphi]\!] \cup indis^E_{\mathrm{A}}([\![\overline{\mathrm{C}}_A\varphi]\!])$.

Next, define three functions on 2^{St}, which fixed points are equal to respectively $[\![\mathrm{EG}\varphi]\!]$, $[\![\mathrm{E}(\varphi \mathrm{U} \psi)]\!]$, and $[\![\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!]$.

1. $\tau_{\mathrm{EG}\varphi}(X) = [\![\varphi]\!] \cap pre(X)$,
2. $\tau_{\mathrm{E}(\varphi \mathrm{U} \psi)}(X) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap pre(X))$,
3. $\tau_{\overline{\mathrm{C}}_A\varphi}(X) = [\![\varphi]\!] \cup indis^E_{\mathrm{A}}(X)$.

Since $\mathrm{EG}\varphi$ is the maximal fixpoint of $\tau_{\mathrm{EG}\varphi}(X)$, it can be computed as $\tau^k_{\mathrm{EG}\varphi}(St)$ for some finite k. Since $\mathrm{E}(\varphi \mathrm{U} \psi)$ is the minimal fixpoint of $\tau_{\mathrm{E}(\varphi \mathrm{U} \psi)}(X)$ it can be computed as $\tau^l_{\mathrm{E}(\varphi \mathrm{U} \psi)}(\emptyset)$ for some finite l. Similarly, for $\tau_{\overline{\mathrm{C}}_A\varphi}(X)$. The above characterization can be now used for defining a model checking algorithm mchk for the formulas of CTLK.

$mchk(M, \varphi)$ {

if $\varphi \in PV$, then return $V^{-1}(\varphi)$,

if $\varphi = \neg\psi$, then return $St \setminus mchk(M, \psi)$,

if $\varphi = \varphi_1 \wedge \varphi_2$, then return $mchk(M, \varphi_1) \cap mchk(M, \varphi_2)$,

if $\varphi = \mathrm{EX}\psi$, then return $mchk_{\mathrm{EX}}(M, \psi)$,

if $\varphi = \overline{\mathrm{K}}_i\psi$, then return $mchk_{\overline{\mathrm{K}}_i}(M, \psi)$,

if $\varphi = \mathrm{EG}\psi$, then return $mchk_{\mathrm{EG}}(M, \psi)$,

if $\varphi = \mathrm{E}(\phi \mathrm{U} \psi)$, then return $mchk_{\mathrm{EU}}(M, \phi, \psi)$,

if $\varphi = \overline{\mathrm{C}}\psi$, then return $mchk_{\overline{\mathrm{C}}}(M, \psi)$.

}

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

$mchk_{\mathrm{EX}}(M,\psi)\{$
$X := mchk(M,\psi);$
$Y := pre(X);$
return Y $\};$

$mchk_{\mathrm{EK}_i}(M,\psi)\{$
$X := mchk(M,\psi);$
$Y := indis_i(X);$
return Y $\};$

$mchk_{\mathrm{EG}}(M,\psi)\{$
$X := mchk(M,\psi);$
$Y := St;$
$Z := \emptyset;$
while $(Z \neq Y)\{$
$Z := Y;$
$Y := X \cap pre(Y)\}$
return Y $\};$

$mchk_{\mathrm{EU}}(M,\psi_1,\psi_2)\{$
$X := mchk(M,\psi_1);$
$Y := mchk(M,\psi_2);$
$Z := \emptyset;$
$W := St;$
while $(Z \neq W)\{$
$W := Z;$
$Z := Y \cup (X \cap pre(Z))\}$
return Z $\};$

$mchk_{\overline{\mathrm{C}}}(M,\psi)\{$
$Y := mchk(M,\psi);$
$Z := \emptyset;$
$W := St;$
while $(Z \neq W)\{$
$W := Z;$
$Z := Y \cup indis^{E}_{\mathrm{A}}(Z))\}$
return Z $\};$

References: The original state labelling algorithm for CTL was introduced in [2]. More information on model checking CTL can be found in [20, 14]. CTLK is treated extensively in [17].

For other non-symbolic approaches to model checking temporal-epistemic properties, one may e.g. refer to [13]. There, a translation of a fragment of LTLK to LTL is proposed, and the SPIN model checker is used for verification.

3.4 Introduction to Symbolic Model Checking

Ordered Binary Decision Diagrams. OBDDs (Ordered Binary Decision Diagrams) are used for succinct representation of Boolean functions. Model checking problem for CTLK can be efficiently encoded into operations on OBDDs. Consider a Boolean function:

$$f : \{0,1\}^n \longrightarrow \{0,1\}$$

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Such a function can be represented by the results of all the valuations of some propositional formula over n propositional variables. For example the function $f(x_1, x_2) = x_1 * x_2$ is represented by the formula $p_1 \wedge p_2$. Each Boolean function can be represented by an OBDD. The size of the BDD is determined both by the function being represented and the chosen ordering of the variables. For a boolean function $f(x_1, \ldots, x_n)$ then depending upon the ordering of the variables we would end up getting a graph whose number of nodes would be linear (in n) at the best and exponential at the worst case.

Let us consider the Boolean function $f(x_1, \ldots, x_{2n}) = x_1x_2 + x_3x_4 + \cdots + x_{2n-1}x_{2n}$. Using the variable ordering $x_1 < x_3 < \cdots < x_{2n-1} < x_2 < x_4 < \cdots < x_{2n}$, the BDD needs 2^{n+1}, nodes to represent the function. Using the ordering $x_1 < x_2 < x_3 < x_4 < \cdots < x_{2n-1} < x_{2n}$, the BDD consists of $2n$ nodes.

The following operations can be implemented by polynomial-time graph manipulation algorithms: disjunction, conjunction, negation, implication, equivalence, existential abstraction, and universal abstraction.

OBBD-Based Model Checking for CTLK. The algorithms from Section 3 (computing, for each formula φ, the set of states $[\![\varphi]\!]$ in which φ holds) can operate on the OBDD representations of the states. This requires to encode the states and the transition relation of a model M by propositional formulas, and then to represent these formulas by OBDDs.

The model checking problem $M, s^0 \models \varphi$ is translated to checking whether $s^0 \in [\![\varphi]\!]$. So, we need to verify whether $OBDD(\{s^0\}) \wedge OBDD([\![\varphi]\!])$ is not equal to $OBDD(\emptyset)$, where $OBDD(S)$ denotes the OBDD representing the set of states S.

References: Our presentation of the OBBD approach to model checking CTLK is based on [21, 18].

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Bibliography

[1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[2] E. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[3] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[4] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[5] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[6] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[7] E. A. Emerson and C-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proc. of the 1st Symp. on Logic in Computer Science (LICS'86)*, pages 267–278. IEEE Computer Society, 1986.

[8] E. A. Emerson and Ch.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

[9] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[10] M. Fisher. *Temporal Logics*. Kluwer, 2006.

[11] J. Y. Halpern. Reasoning about knowledge: a survey. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *The Handbook of Logic in Artificial Intelligence and Logic Programming, Volume IV*, pages 1–34. Oxford University Press, 1995.

[12] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[13] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of the 9th Int. SPIN Workshop (SPIN'02)*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.

[14] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

[15] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[16] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL '85: Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 97–107, New York, NY, USA, 1985. ACM.

[17] A. Lomuscio and W. Penczek. Logic column 19: Symbolic model checking for temporal-epistemic logics. *CoRR*, abs/0709.0446, 2007.

[18] A. Lomuscio, H. Qu, and F. Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In *CAV*, pages 682–688, 2009.

[19] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley : Reading, 1994.

[20] W. Penczek and A. Polrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*, volume 20 of *Studies in Computational Intelligence*. Springer-Verlag, 2006.

[21] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *AAMAS*, pages 630–637, 2004.

[22] A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1991.

[23] Ph. Schnoebelen. The complexity of temporal model checking. In *Advances in Modal Logics, Proceedings of AiML 2002*. World Scientific, 2003.

[24] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of ACM*, 32(3):733–749, 1985.

[25] C. Stirling. *Modal and Temporal Properties of Processes*. Springer-Verlag, 2001.

[26] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[27] W. van der Hoek and R. Verbrugge. Epistemic logic: A survey. *Game Theory and Applications*, 8:53–94, 2002.

[28] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science (LICS 1986)*, pages 332–344. IEEE Computer Society Press, 1986.

[29] G. Weiss, editor. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press: Cambridge, Mass, 1999.

[30] M. Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons, 2002.

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY