TU Clausthal
Clausthal University of Technology

Model Checking Temporal and Strategic Logics

Nils Bulling and Jürgen Dix

EASSS 2010
École Nationale Supérieure des Mines
Saint-Étienne, France

23–27 August 2010

TU Clausthal
Clausthal University of Technology

Time

Duration: 6 hours
Thursday, 26. August: 16:30–18:30,
Friday, 27. August: 11:00–13:00 ,16:30–18:30

Course type

Level: advanced
Prerequisites: knowledge of propositional logic, basics of automata and complexity theory

Course website

`http://cig.in.tu-clausthal.de/easss2010`

TU Clausthal
Clausthal University of Technology

About this course

In Part I of this lecture we introduce *modal logic* and show how it can be used to reason about *knowledge of agents.* We also discuss the *correspondence* of axioms with semantic properties of the models.

In Part II we consider the *temporal logics* **LTL, CTL** and **CTL***. We discuss in Part III the *model checking problem* and elaborates on the complexity needed to solve it for the considered temporal logics. We follow an *automata theoretic approach*.

In Part IV we introduce *strategic logics* which are used to model the abilities of agents. We discuss the effect of perfect and imperfect information and perfect and imperfect recall. Finally, in Part V, we consider the model checking problems of these logics.

TU Clausthal
Clausthal University of Technology

Course Overview

Part I: *Modal Logics and Agents*
90 minutes

Part II: *Temporal Logics*
60 minutes

Part III: *Model Checking I: Temporal Logics*
90 minutes

Part IV: *Strategic Logics*
60 minutes

Part V: *Model Checking II: Strategic Logics*
60 minutes

TU Clausthal
Clausthal University of Technology

Reading Material I

- Alur, R., Henzinger, T. A., and Kupferman, O. (2002).
  Alternating-time Temporal Logic.
  Journal of the ACM, 49:672–713.
- Baier, C. and Katoen, J.-P. (2008).
  Principles of Model Checking.
  The MIT Press.
- Blackburn, P., de Rijke, M., and Venema, Y. (2001).
  Modal Logic.
  Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK.

TU Clausthal
Clausthal University of Technology

Reading Material II

- Blackburn, P., Benthem, J. F. A. K. v., and Wolter, F. (2006a).
  Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning).
  Elsevier Science Inc., New York, NY, USA.
- Bulling, N., Dix, J., and Jamroga, W. (2010).
  Model checking logics of strategic ability: Complexity.
  In Dastani, M., Hindriks, K. V., and Meyer, J.-J. C., editors, *Specification and Verification of Multi-Agent Systems*. Springer.

TU Clausthal
Clausthal University of Technology

Reading Material III

- Clarke, E., Grumberg, O., and Peled, D. (1999).
  Model Checking.
  MIT Press.
- Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. (1995).
  Reasoning about Knowledge.
  MIT Press: Cambridge, MA.
- Schnoebelen, P. (2003).
  The complexity of temporal model checking.
  In *Advances in Modal Logics, Proceedings of AiML 2002*. World Scientific.

TU Clausthal
Clausthal University of Technology

Outline

1. Modal Logics and Agents

1.1 Propositional Logic

Syntax

- The *propositional language* is built upon
  - *Propositional symbols*: $p, q, r, \ldots, p_1, p_2, p_3, \ldots$
  - *Logical connectives*: $\neg$ and $\vee$
  - *Grouping symbols*: $(,)$
- Often we consider only a finite, nonempty set of propositional symbols and refer to it as $\mathcal{P}rop$.
- *Propositional language* $\mathcal{L}_{PL}(\mathcal{P}rop)$:
  $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi$
- *Macros*:

$$\begin{array}{rcl} \bot & := & p \wedge \neg p \\ \top & := & \neg\bot \end{array} \qquad \begin{array}{rcl} \varphi \wedge \psi & := & \neg(\neg\varphi \vee \neg\psi) \\ \varphi \rightarrow \psi & := & \neg\varphi \vee \psi \\ \varphi \leftrightarrow \psi & := & (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \end{array}$$

Semantics

- A *valuation* (or *truth assignment*) $v : \mathcal{P}rop \rightarrow \{\mathbf{t}, \mathbf{f}\}$ for a language $\mathcal{L}_{PL}(\mathcal{P}rop)$ is a *mapping* from the *set of propositional constants* defined by $\mathcal{P}rop$ into the set $\{\mathbf{t}, \mathbf{f}\}$.
- Inductively, we define the notion of a formula φ being *true* or *satisfied* by v (denoted by $v \models \varphi$):
  $v \models p$ iff $v(p) = \mathbf{t}$ and $p \in \mathcal{P}rop$,
  $v \models \neg\varphi$ iff not $v \models \varphi$,
  $v \models \varphi \vee \psi$ iff $v \models \varphi$ or $v \models \psi$
- For a set $\Sigma \subseteq \mathcal{L}_{PL}$ we write $v \models \Sigma$ iff $v \models \varphi$ for all $\varphi \in \Sigma$.
- We use $v \not\models \varphi$ instead of not $v \models \varphi$.

Truth Tables

Truth tables are a conceptually simple way of working with PL (invented by Wittgenstein in 1918).

p	q	$\neg p$	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$p \leftrightarrow q$
t	t	f	t	t	t	t
f	t	t	t	f	t	f
t	f	f	t	f	f	f
f	f	t	f	f	t	t

Fundamental Semantical Concepts

- If it is possible to find *some valuation* v that makes φ true, then we say φ is *satisfiable*.
- If $v \models \varphi$ *for all valuations* v then we say that φ is *valid* and write $\models \varphi$. φ is also called *tautology*.
- A *theory* is a set of formulae: $T \subseteq \mathcal{L}_{PL}$.
- A theory T is called *consistent* if *there is a valuation* v *with* $v \models T$.
- A theory T is called *complete* if for each formula φ in the language, $\varphi \in T$ *or* $\neg\varphi \in T$.

Two simple examples

- Is $p \wedge \neg b$ *satisfiable*?
- Is $a \vee \neg a$ *valid*?

Consequences

Given a theory T we are interested in the following question: **Which facts can be derived from T?** We can distinguish two approaches:

1. *semantical* consequences, and
2. *syntactical* inference.

- Let T be a *theory* and φ be a formula. We say that φ is a *semantical consequence of* T if *for all valuations* v:
  $v \models T$ implies $v \models \varphi$.
- If new facts can be derived from old ones in an **algorithmic** fashion, we say that such a fact is *syntactically derivable*.
- *Inference rules* allow to derive new facts:

$$(MP) \quad \frac{\varphi \qquad \varphi \rightarrow \psi}{\psi}$$

1.2 Basic Modal Logic

What is a Logic?

We present a *framework for thinking about logics as*:

- *languages* for describing a problem,
- ways of talking about *relational structures* and *models*.

These are the two key components in the way we will approach logic:

1. *Language:*
   fairly simple, precisely defined, formal languages.
2. *Model* (or *relational structure*):
   simple "world" that the logic talks about.

Various modal logics

- knowledge → **epistemic logic**,
- beliefs → **doxastic logic**,
- obligations → **deontic logic**,
- actions → **dynamic logic**,
- time → **temporal logic**,
- and *combinations of the above*.

Most famous **multimodal logics**:
BDI logics of beliefs, desires, intentions (and time).

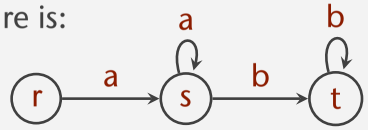
Relational Structures

A *relational structure* is given by $(W, \{\mathcal{R}_1, \ldots, \mathcal{R}_n\})$ and consists of:

- A *non-empty* set W, the elements of which are our *objects* of interest. They are called *points, states, nodes, worlds, times, instants or situations*.
- A *non-empty* set $\{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ of *relations*, $\mathcal{R}_i \subseteq W \times W$.

Example 1.1 (Finite state automaton for $a^n b^m$)

Given the formal language $a^n b^m$ with $n, m > 0$. The relational structure is:



Here, r is the start state and t is the only final state.

The Basic Modal Language

- *Standard propositional logic* can be seen as a *one-point relational structure*.
- But relational structures can describe much more. We can talk about *points*, *lines* etc.
- Therefore, we introduce the *basic modal language*.

We build the *basic modal language* on top of the *propositional language* by extending $\mathcal{L}_{PL}(\mathcal{P}rop)$ with two new operators:

Possibility and necessity

$\Diamond\varphi$: φ is *possible*
(We see one or more states where φ holds.)

$\Box\varphi$: φ is *necessary*
(In all reachable states φ holds.)

A Language for Relational Structures

Definition 1.2 (Basic modal language $\mathcal{L}_{\mathsf{BML}}$)

Let $\mathcal{P}rop$ be a set of propositions. The *basic modal language* $\mathcal{L}_{\mathsf{BML}}(\mathcal{P}rop)$ consists of all formulae defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond\varphi$$

where $p \in \mathcal{P}rop$.

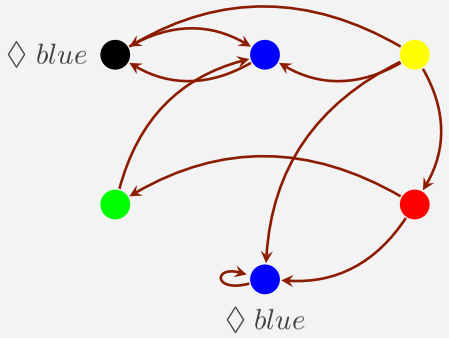
Boolean macros are defined in the standard way. Additionally, we have the dual $\Box$ (called "*box*") of $\Diamond$:

$$\Box\varphi := \neg\Diamond\neg\varphi$$

We can talk about attributes by **adding labels** to nodes (e.g. painting them in a particular color).

Example 1.3 (Colored graph I)

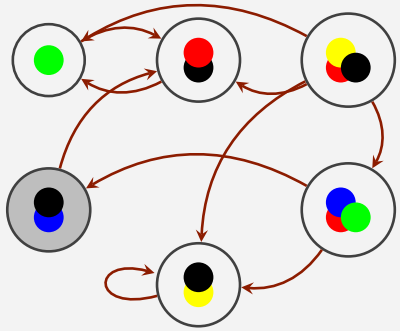
Imagine standing in a node of a colored graph. What can we see?



Example 1.4 (Colored graph II)

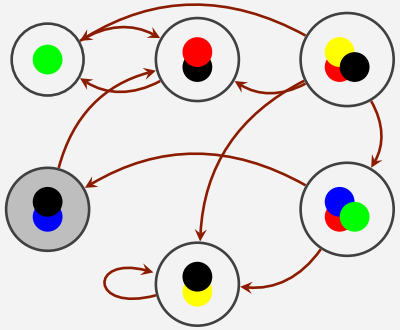
We imagine standing in a node of a colored graph. What can we see?

$$\Diamond(black \wedge red) \wedge \Diamond\Diamond green$$



Colored graph II

Example 1.5



$blue \longrightarrow \Box black$

$green \longrightarrow \Box black$

$yellow \longrightarrow \Diamond yellow$

Definition 1.6 (Kripke frame)

A *Kripke frame* is given by $\mathfrak{F} = (W, \mathcal{R})$ where

- W is a non-empty set, called set of **domains** or **worlds**,
- $\mathcal{R} \subseteq W \times W$ is a binary relation.

Frames are mainly used to talk about *validities*: They stand for a whole *set of models*.

Definition 1.7 (Kripke model)

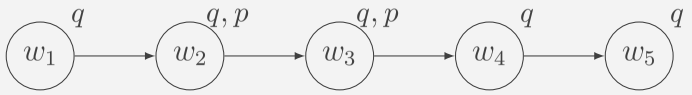
A *Kripke model* is given by $\mathfrak{M} = (W, \mathcal{R}, V)$ where

- $(W, \mathcal{R})$ is a *Kripke frame*,
- $V : \mathcal{P}rop \to \mathcal{P}(W)$ is called *labelling function* or *valuation*. We also use $V : W \to \mathcal{P}(\mathcal{P}rop)$.

Kripke frames (resp. models) are simply *relational structures* (resp. with labels)!

Example 1.8

Consider the frame $\mathfrak{F} = (\{w_1, w_2, w_3, w_4, w_5\}, \mathcal{R})$ where $\mathcal{R}w_iw_j$ iff $j = i + 1$ and $V(p) = \{w_2, w_3\}$, $V(q) = \{w_1, w_2, w_3, w_4, w_5\}$, $V(r) = \emptyset$.



Frames vs. Models?

Frames

Mathematical pictures of ontologies that we find interesting. That is, frames define the *fundamental structure* of the domain of interest.

For example, we model *time* as a *collection of points ordered by a strict partial order*.

Models

Frames are extended by *contingent* information. That is, models extend the mathematical structure provided by frames by *additional information*.

Can Kripke models be used to interpret the propositional language?

Formal semantics of $\mathcal{L}_{\mathsf{ML}}$.

Definition 1.9 (Semantics $\mathfrak{M}, w \models \varphi$)

Let $\mathfrak{M}$ be a Kripke model, $w \in W_{\mathfrak{M}}$, and $\varphi \in \mathcal{L}_{\mathsf{ML}}$. φ is said to be *locally true* or *satisfied in $\mathfrak{M}$ and world w*, written as $\mathfrak{M}, w \models \varphi$, if the following holds:

$\mathfrak{M}, w \models p$ iff $w \in V_{\mathfrak{M}}(p)$ and $p \in \mathcal{P}rop$,

$\mathfrak{M}, w \models \neg\varphi$ iff *not* $\mathfrak{M}, w \models \varphi$

$\mathfrak{M}, w \models \varphi \vee \psi$ iff $\mathfrak{M}, w \models \varphi$ *or* $\mathfrak{M}, w \models \psi$

$\mathfrak{M}, w \models \Diamond\varphi$ iff *there is* a world $w' \in W$ such that Rww' and $\mathfrak{M}, w' \models \varphi$

Given a set $\Sigma \subseteq \mathcal{L}_{\mathsf{ML}}$ we write $\mathfrak{M}, w \models \Sigma$ iff $\mathfrak{M}, w \models \varphi$ *for all* $\varphi \in \Sigma$.

What about $\Box\varphi$? $\rightsquigarrow$ *blackboard*

Internal and Local

Satisfaction of formulae is *internal* and *local*!

Internal: Formulae are evaluated *inside* models at some *given world*.

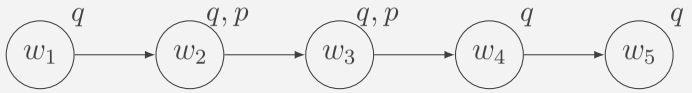
Local: Given a world it is only possible to refer to *direct sucessors* of this world.

How does first-order logic compare to that?

Some Examples

Example 1.10

$\mathfrak{F} = (\{w_1, w_2, w_3, w_4, w_5\}, \mathcal{R})$ where $\mathcal{R}w_iw_j$ iff $j = i + 1$ and $V(p) = \{w_2, w_3\}$, $V(q) = \{w_1, w_2, w_3, w_4, w_5\}$, $V(r) = \emptyset$.



1. $\mathfrak{M}, w_1 \models \Diamond\Box p$
2. $\mathfrak{M}, w_1 \not\models \Diamond\Box p \rightarrow p$
3. $\mathfrak{M}, w_2 \models \Diamond(p \wedge \neg r)$
4. $\mathfrak{M}, w_1 \models q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond q))))$
5. $\mathfrak{M} \models \Box q$

Validity and (Global) Satisfaction

We take on a *global point of view*.

Given a *specification* like $\varphi := \Box\neg crash$. In which states should it be true?

Definition 1.11 (Validity)

A formula φ is called *valid* or *globally true* in a model $\mathfrak{M}$ iff $\mathfrak{M}, w \models \varphi$ *for all* $w \in W_{\mathfrak{M}}$. We write $\mathfrak{M} \models \varphi$.

φ is *satisfiable* in $\mathfrak{M}$ if $\mathfrak{M}, w \models \varphi$ *for some* $w \in W_{\mathfrak{M}}$.

Analogously, we say that a *set Σ of formulae* is *valid* (resp. *satisfiable*) in $\mathfrak{M}$ iff *all* formulae in Σ are valid (resp. satisfiable) in $\mathfrak{M}$.

Validity and satisfiability are *dual concepts*!

Example 1.12

In which models is the following formula true?

$$\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$$

$\mathfrak{M}, w \models \Box(p \rightarrow q)$
iff $\forall w'(w\mathcal{R}w' \Rightarrow \mathfrak{M}, w' \models p \rightarrow q)$
iff $\forall w'(w\mathcal{R}w' \Rightarrow (\mathfrak{M}, w' \models p \Rightarrow \mathfrak{M}, w' \models q))$
implies $\forall w'(w\mathcal{R}w' \Rightarrow \mathfrak{M}, w' \models p) \Rightarrow$
$\forall w'(w\mathcal{R}w' \Rightarrow \mathfrak{M}, w' \models q)$
iff $\mathfrak{M}, w \models \Box p \Rightarrow \mathfrak{M}, w \models \Box q$
iff $\mathfrak{M}, w \models \Box p \rightarrow \Box q$

The formula is true in *any frame* and hence in **any model**.

Modal Consequence Relation

Up to now we verified formulae in a given model and state. Often, it is interesting to know whether a property *follows* from a given set of formulae.

Definition 1.13 (Local Consequence Relation)

Let $\mathcal{M}$ be a *class of models*, Σ be a *set of formulae* and φ be a formula.

- φ is a *(local) semantic consequence of* Σ over $\mathfrak{M}$, written $\Sigma \models_{\mathcal{M}} \varphi$, if *for all* $\mathfrak{M} \in \mathcal{M}$ *and all* $w \in W_{\mathfrak{M}}$ it holds that $\mathfrak{M}, w \models \Sigma$ *implies* $\mathfrak{M}, w \models \varphi$.
- If $\mathcal{M}$ is the class of *all* models we just say that φ is a (local) consequence of Σ and write $\Sigma \models \varphi$.

$\rightsquigarrow$ *blackboard*

Frames and Validity

In Example 1.12 we have seen that a formula can be true/false for all valuations. We can speak about *structural properties* ignoring contingent information.

Definition 1.14 (Frame Validity: $\mathfrak{F} \models \varphi$)

Let $\mathfrak{F}$ be a frame and $\varphi \in \mathcal{L}_{\text{BML}}$.

1. φ is *valid in* $\mathfrak{F}$ *and* $w \in W_{\mathfrak{F}}$, written $\mathfrak{F}, w \models \varphi$, if $\mathfrak{M}, w \models \varphi$ *for all models* $\mathfrak{M} = (\mathfrak{F}, \pi)$ *based on* $\mathfrak{F}$.
2. φ is *valid in* $\mathfrak{F}$, written $\mathfrak{F} \models \varphi$, if $\mathfrak{F}, w \models \varphi$ for all $w \in W_{\mathfrak{F}}$.
3. Let $\mathcal{F}$ be *class* of frames. φ is said to be *valid in* $\mathcal{F}$, if φ is valid in *each frame* $\mathfrak{F} \in \mathcal{F}$.

Lemma 1.15 (Distribution Axioms)

The two formulae

$$\Diamond(p \vee q) \rightarrow (\Diamond p \vee \Diamond q)$$
$$\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$$

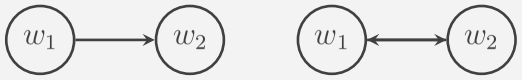
are both valid in all Kripke frames $\mathfrak{F}$. *The last formula is also called axiom* ***K***.

Proof.

$\rightsquigarrow$ Exercise and Example 1.12. □

Example 1.16

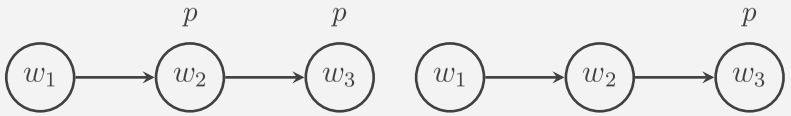
Is $\Diamond\top$ valid in *all frames*? In which class is the formula valid?



What about $\Box\top$? $\rightsquigarrow$ *blackboard*

Example 1.17

Is $\Diamond\Diamond p \rightarrow \Diamond p$ true in w_1?



Is there a *class of frames* in which formula is *valid*? $\rightsquigarrow$ *blackboard*

Example 1.18

Let $\mathcal{M}$ be the **class of transitive models**. Then:

1. $\Diamond\Diamond p \models_{\mathcal{M}} \Diamond p$,
2. $\Box p \models_{\mathcal{M}} \Box\Box p$, but
3. $\Box\Box p \models_{\mathcal{M}} \Box p$ does not hold.

Is there a class of *models* $\mathcal{M}$ for which $\Diamond\Diamond p \models_{\mathcal{M}} \Diamond p$ holds, but **no model in $\mathcal{M}$ is transitive**?

1.3 Correspondence Theory

Correspondence Theory

We have learnt that some formulae are valid in particular frames. E.g. $\Diamond\Diamond\varphi \rightarrow \Diamond\varphi$ is valid in all *transitive frames*. Here, we consider such *correspondences systematically*.

Definition 1.19 (KDT45)

We assume that we have available one modal operator $\Box$.

K	$\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$
D	$\neg\Box(p \wedge \neg p)$
T	$\Box p \rightarrow p$
4	$\Box p \rightarrow \Box\Box p$
5	$\neg\Box p \rightarrow \Box\neg\Box p$

In *epistemic logic*, e.g., these formulae will have intuitive epistemic properties.

Properties of Frame (1)

We consider properties of the *accessibility relations* $\mathcal{R}$ of frames:

Serial: For all w there is a w' with $w\mathcal{R}w'$.
Reflexive: For all w: $w\mathcal{R}w$.
Transitive: For all w, w', w'': $w\mathcal{R}w'$ and $w'\mathcal{R}w''$ implies $w\mathcal{R}w''$.
Euclidean: For all w, w', w'': $w\mathcal{R}w'$ and $w\mathcal{R}w''$ implies $w'\mathcal{R}w''$.
Symmetric: For all w, w': $w\mathcal{R}w'$ implies $w'\mathcal{R}w$.

Definition 1.20 (Frame property)

We say a *frame* $\mathfrak{F} = (W, \mathcal{R})$ *has property* X if its relation $\mathcal{R}$ has property X.

Remember Slide 37 where we discussed *transitive frames* .

Example 1.21

We have

$$\mathfrak{F} \models \Box p \to p \text{ iff } \mathfrak{F} \text{ is } \textit{reflexive}.$$

Let $\mathfrak{F}$ be a frame satisfying $\Box p \to p$. That is,

$$\text{for all } w \in W, \mathfrak{F}, w \models \Box p \to p.$$

This is the case, if for all *models* $\mathfrak{M}$ over $\mathfrak{F}$ and

$$\text{for all } w \in W, \mathfrak{M}, w \models \Box p \to p.$$

Which properties must $\mathcal{R}$ satisfy? Suppose $\mathcal{R}$ is *not reflexive*. Then, there is a state w' with *not* $w'\mathcal{R}w'$. *Make* p *true* at all states of $W \backslash \{w'\}$. Then, $\mathfrak{M}, w' \not\models \Box p \to p$ and hence $\mathfrak{F} \not\models \Box p \to p$. *Contradiction!* $\rightsquigarrow$ *blackboard*

Now suppose we are given a *reflexive frame* $\mathfrak{F}$ and suppose $\mathfrak{F} \not\models \Box p \to p$.

- Then, there is a model $\mathfrak{M} = (\mathfrak{F}, \pi)$ and a state w, $\mathfrak{M}, w \not\models \Box p \to p$.
- That is, $\mathfrak{M}, w \models \Box p$ and $\mathfrak{M}, w \not\models p$.
- By reflexivity we have $w\mathcal{R}w$.
- But then, from $\mathfrak{M}, w \models \Box p$ it follows that $\mathfrak{M}, w \models p$. *Contradiction!*
- We must have $\mathfrak{F} \models \Box p \to p$.

In other words, axiom T *characterises reflexive frames*.

Validity in Several Frames (3)

Lemma 1.22 (Appropriate Frames)

Let $(W, \mathcal{R})$ *be a Kripke frame. Then the following holds:*

- **K:** $(W, \mathcal{R}) \models \Box(p \to q) \to (\Box p \to \Box q)$.
- **D:** $(W, \mathcal{R}) \models \neg\Box(p \wedge \neg p)$ *iff* $\mathcal{R}$ *is serial.*
- **T:** $(W, \mathcal{R}) \models \Box p \to p$ *iff* $\mathcal{R}$ *is reflexive.*
- **4:** $(W, \mathcal{R}) \models \Box p \to \Box\Box p$ *iff* $\mathcal{R}$ *is transitive.*
- **5:** $(W, \mathcal{R}) \models \neg\Box p \to \Box\neg\Box p$ *iff* $\mathcal{R}$ *is Euclidean.*
- **B:** $(W, \mathcal{R}) \models p \to \Box\Diamond p$ *iff* $\mathcal{R}$ *is symmetric.*

Proof.

$\rightsquigarrow$: Exercise. □

Axiomatic Systems

As in classical logic, one can ask about a **complete** axiom system. Is there a calculus that allows to derive all sentences *true in all Kripke models*?

Definition 1.23 (System K)

The *system* ***K*** is an *extension* of the *propositional calculus* by the axiom

$$K \quad (\Box\varphi \wedge \Box(\varphi \to \psi)) \to \Box\psi$$

and the *inference rule* $\dfrac{\varphi}{\Box\varphi}$ (*Necessitation*).

We also need the *duality axiom* $\Box\varphi \leftrightarrow \neg\Diamond\neg\varphi$ (as we introduced $\Box$ as macro).

Note, φ and ψ can be *substituted* by any formula.

Proposition 1.24

Axiom K is equivalent to $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$.

Example 1.25

Proof that $\vdash_K (\Box p \wedge \Box q) \to \Box(p \wedge q)$. $\rightsquigarrow$ *blackboard*

Theorem 1.26 (Sound-/completeness of K)

System ***K*** *is sound and complete with respect to* **arbitrary Kripke models**.

- Note that we have *not assumed any properties* of the accessibility relation $\mathcal{R}$: It is just *any* binary relation.
- Assuming that $\mathcal{R}$ is an *equivalence* relation, what *additional statements* (axioms) are true in all such Kripke models?

Theorem 1.27 (Sound/complete subsystems)

Let $\mathbf{X}$ *be any subset of* $\{\mathbf{D}, \mathbf{T}, \mathbf{4}, \mathbf{5}\}$ *and let* $\mathcal{X}$ *be the subset of* $\{$*serial, reflexive, transitive, euclidean*$\}$ *corresponding to* $\mathbf{X}$. *Then system* ***K*** *extended with axioms* $\mathbf{X}$ *is sound and complete with respect to Kripke frames which satisfy properties* $\mathcal{X}$.

For example, we have the following important instance:

Corollary 1.28 (KT45)

System ***KT45*** *is sound and complete with respect to Kripke frames with an accessibility relation which is an equivalence relation.*

Example 1.29

KT45 *and* **KTD45** are both *sound and complete* with respect to Kripke frames in which the accessibility relation is an *equivalence relation*.

What does that mean for the properties corresponding to these axioms?

Any reflexive, transitive and euclidean relation is also serial!

Some Exercises

Show that

1. The axiom **D** follows from **KT45**.
2. Show that **KD45** is not equivalent to **K45**: axiom **D** does not follow from **K45**.

One can argue *semantically*:

- **Each** Kripke model satisfying the axioms **KT45**, also satisfies the axiom **D**.
- **There is** a Kripke model satisfying **K45** in which **D** does not hold.

1.4 Epistemic Logic

Interpreting $\Box_i$ as knowledge

Let us now assume we have *several agents* i and we interpret $\Box_i\varphi$ as *agent* i *knows that* φ. In that case one often writes

$\mathbf{K_i}\varphi$ instead of $\Box_i\varphi$.

Accessibility relation

What does the *equivalence relation* encode? *Incomplete information*:

$w\mathcal{R}w' \rightsquigarrow$ The agent *cannot distinguish* w and w'. Both states *provide the same information*.

Knowledge = *Truth in all indistinguishable states*
$\rightsquigarrow$ *blackboard*

What other properties should hold when interpreting $\Box$ as knowledge?

K	$K(p \rightarrow q) \rightarrow (Kp \rightarrow Kq)$	
D	$\neg K\bot$	$\rightsquigarrow$ consistency
T	$Kp \rightarrow p$	$\rightsquigarrow$ truth
4	$Kp \rightarrow KKp$	$\rightsquigarrow$ positive introspection
5	$\neg Kp \rightarrow K\neg Kp$	$\rightsquigarrow$ negative introspection

Muddy children

Example 1.30 (Muddy children)

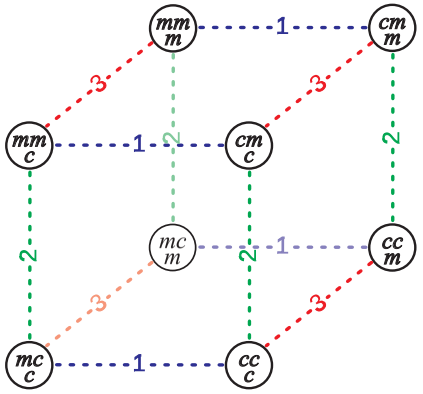
- A *group of playing children* is called back by their father. They *gather around him*.
- *Some of them have become dirty*:
  1. *they may have mud on their forehead*,
  2. *children can only see whether others are muddy*,
  3. *and not if there is any mud on their own forehead*.
- All this is *commonly known*, and the children *are perfect logicians*.

Now the father announces the following:

- Father: *"At least one of you has mud on his or her forehead."*
- Father: *"Will those who know whether they are muddy please step forward."*
- If nobody steps forward, father *keeps repeating the request*.

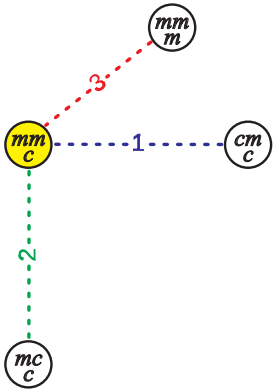
Question

What happens?

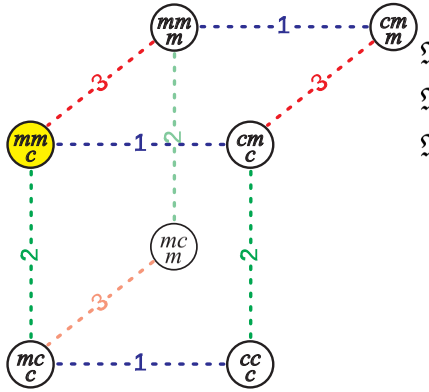


$m_1 \in V(w)$ iff $w = q_{mxx}$
$m_2 \in V(w)$ iff $w = q_{xmx}$
$m_3 \in V(w)$ iff $w = q_{xxm}$

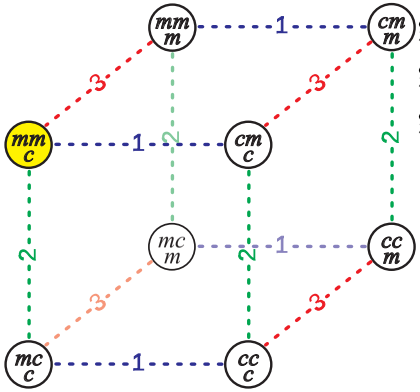
$\mathfrak{M}, q_{mmc} \models m_1 \wedge m_2 \wedge \neg m_3$



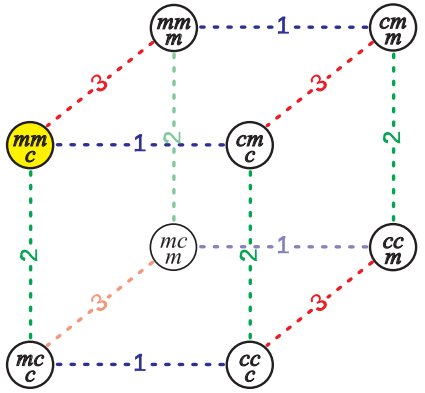
$$\mathfrak{M}, q_{mmc} \models m_1 \wedge m_2 \wedge \neg m_3$$
$$\mathfrak{M}, q_{mmc} \models \neg K_1 m_1 \wedge K_1 m_2$$



$$\mathfrak{M}, q_{mmc} \models m_1 \wedge m_2 \wedge \neg m_3$$
$$\mathfrak{M}, q_{mmc} \models \neg K_1 m_1 \wedge K_1 m_2$$
$$\mathfrak{M}, q_{mmc} \models K_1 K_3 m_2 \wedge K_1 \neg K_2 m_2$$

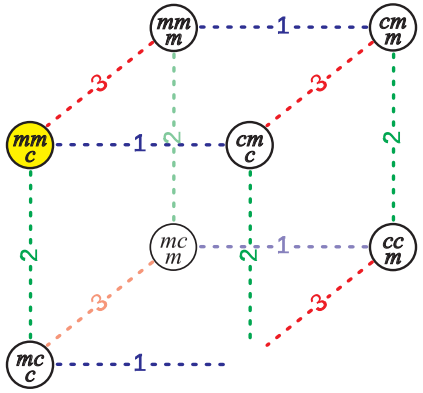


$$\mathfrak{M}, q_{mmc} \models m_1 \wedge m_2 \wedge \neg m_3$$
$$\mathfrak{M}, q_{mmc} \models \neg K_1 m_1 \wedge K_1 m_2$$
$$\mathfrak{M}, q_{mmc} \models K_1 K_3 m_2 \wedge K_1 \neg K_2 m_2$$
$$\ldots$$



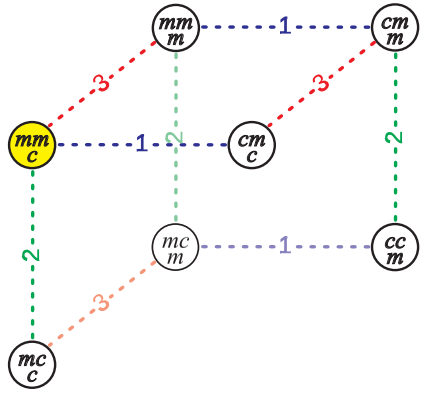
Father: "At least one is muddy."

$\neg K_1 m_1 \wedge \neg K_2 m_2 \wedge \neg K_3 m_3$



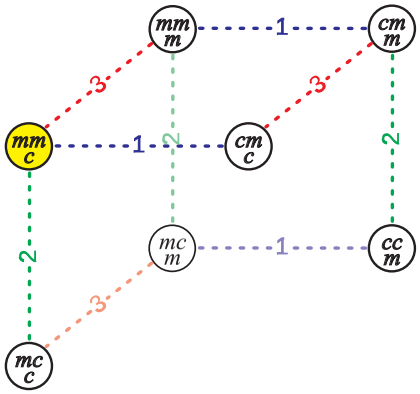
Father: "At least one is muddy."

$\neg K_1 m_1 \wedge \neg K_2 m_2 \wedge \neg K_3 m_3$

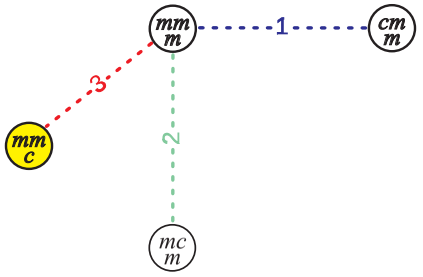


Father (1): "If you know that you're muddy, raise your hand."

Nothing happens.



- Father (2): "If you know that you're muddy, raise your hand."
- The kids see that *nobody* has raised their hands after (1)!
- Children with mud can *eliminate worlds*...



- Father (2): "If you know that you're muddy, raise your hand."
- The kids see that *nobody* has raised their hands after (1)!
- Children with mud can *eliminate worlds*...
- What happens?
- $\rightsquigarrow$ *Kids 1 and 2 raise their hands*:
  $K_1 m_1 \wedge K_2 m_2 \wedge \neg K_3 m_3$

Interpreting $\Box$ as *belief*

Up to now we were thinking of $\Box_i$ as *agent i knows that φ*.
What if we interpret the operator as *belief*?

Under such an interpretation axiom **T** is usually not assumed to hold. But all other axioms make sense.

Definition 1.31 (System KD45)

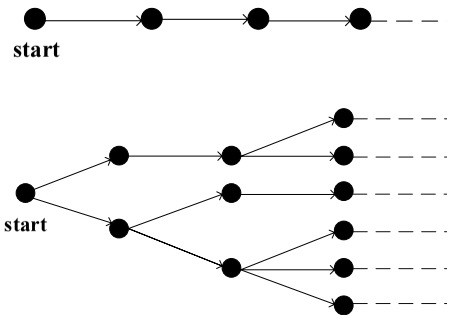
Axiom system $\mathbf{KD45}$ is called the *standard logic of beliefs*. Axiom **K** is called logical omniscience, axiom **D** is called consistency, axiom **4** (resp. axiom **5**) is called positive (resp. negative) introspection.

2. Temporal Logics

2.1 Linear-Time Logic

Reasoning about Time

- The *accessibility relation* represents *time*.
- Time: *linear* vs. *branching*.



Temporal logic was originally developed in order to *represent tense in natural language.*

Within Computer Science, it has achieved a significant role in the **formal specification and verification of concurrent and distributed systems**.

Much of this *popularity* has been achieved because a number of *useful concepts can be formally*, and concisely, specified using temporal logics, e.g.

- *safety properties*
- *liveness properties*
- *fairness properties*

Typical temporal operators

$\bigcirc\varphi$	φ is true in the *next* moment in time
$\Box\varphi$	φ is true in *all* future moments
$\Diamond\varphi$	φ is true in *some* future moment
$\varphi\,\mathcal{U}\,\psi$	φ is true *until* the moment when ψ becomes true

$$\Box((\neg passport \vee \neg ticket) \rightarrow \bigcirc \neg board\_flight)$$
$$send(msg, rcvr) \rightarrow \Diamond receive(msg, rcvr)$$

Safety Properties

"something bad will not happen"
"something good will always hold"

Typical examples:

$\Box\neg bankrupt$
$\Box fuelOK$
and so on ...

Usually: $\Box\neg$....

Liveness Properties

"something good will happen"

Typical examples:

$\Diamond rich$
$power\_on \rightarrow \Diamond online$
and so on ...

Usually: $\Diamond$....

Fairness Properties

Combinations of safety and liveness possible:

$\Diamond\Box dead$

$\Box(request\_taxi \rightarrow \Diamond arrive\_taxi)$ $\rightsquigarrow$ *fairness*

Strong fairness

"If something is *requested* then it will be *allocated*":

$\Box(attempt \rightarrow \Diamond success)$,
$\Box\Diamond attempt \rightarrow \Box\Diamond success$.

- Scheduling processes, responding to messages, etc.
- No process is blocked forever, etc.

Linear-Time Temporal Logic

- Reasoning about a *particular computation* of a system.
- Time is *linear*: just one possible future moment!
- *Models*: paths (e.g. obtained from Kripke structures)

$$\lambda : \mathbb{N}_0 \rightarrow Q.$$

Definition 2.1 (Language $\mathcal{L}_{LTL}$ [Pnueli, 1977])

The *language* $\mathcal{L}_{LTL}(\mathcal{P}rop)$ is given by all formulae generated by the following grammar, where $p \in \mathcal{P}rop$ is a proposition:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \bigcirc\varphi.$$

The additional operators

- $\Diamond$ (*now or sometime from now on*) and
- $\Box$ (*always from now on*)

can be defined as *macros* :

$$\Diamond\varphi \equiv \top \,\mathcal{U}\, \varphi \quad \text{and} \quad \Box\varphi \equiv \neg\Diamond\neg\varphi$$

$\rightsquigarrow$ *blackboard*

The standard Boolean connectives $\top, \bot, \vee, \rightarrow$, and $\leftrightarrow$ are defined in their usual way.

Models of LTL

The semantics is given over *paths*, which are **infinite sequences of states** from Q, and a standard *labelling function* $\pi : Q \rightarrow \mathcal{P}(\mathcal{P}rop)$ that determines which *propositions are true* at which states.

Definition 2.2 (Path λ)

- A *path* λ over a set of states Q is an *infinite sequence* from Q^ω. We also identify it with a *mapping* $\mathbb{N}_0 \rightarrow Q$.
- We use
  - $\lambda[i]$ *to denote the* i*th position* on path λ (starting from $i = 0$) and
  - $\lambda[i, \infty]$ *to denote the subpath of* λ *starting from* i ($\lambda[i, \infty] = \lambda[i]\lambda[i+1]\ldots$).

$$\lambda = q_1 q_2 q_3 \ldots \in Q^\omega$$

Definition 2.3 (Semantics of LTL)

Let λ be a *path* and π be a *labelling function* over Q. The semantics of LTL, $\models^{LTL}$, is defined as follows:

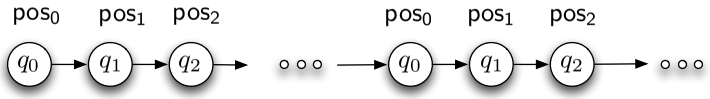
- $\lambda, \pi \models^{LTL} p$ iff $p \in \pi(\lambda[0])$ and $p \in \mathcal{P}rop$;
- $\lambda, \pi \models^{LTL} \neg\varphi$ iff *not* $\lambda, \pi \models^{LTL} \varphi$ (we will also write $\lambda, \pi \not\models^{LTL} \varphi$);
- $\lambda, \pi \models^{LTL} \varphi \wedge \psi$ iff $\lambda, \pi \models^{LTL} \varphi$ *and* $\lambda, \pi \models^{LTL} \psi$;
- $\lambda, \pi \models^{LTL} \bigcirc\varphi$ iff $\lambda[1, \infty], \pi \models^{LTL} \varphi$; and
- $\lambda, \pi \models^{LTL} \varphi \mathcal{U} \psi$ iff *there is an* $i \in \mathbb{N}_0$ such that $\lambda[i, \infty], \pi \models \psi$ and $\lambda[j, \infty], \pi \models^{LTL} \varphi$ *for all* $0 \leq j < i$.

Other temporal operators

$\lambda, \pi \models \Diamond\varphi$ iff $\lambda[i, \infty], \pi \models \varphi$ for *some* $i \in \mathbb{N}_0$;
$\lambda, \pi \models \Box\varphi$ iff $\lambda[i, \infty], \pi \models \varphi$ *for all* $i \in \mathbb{N}_0$;

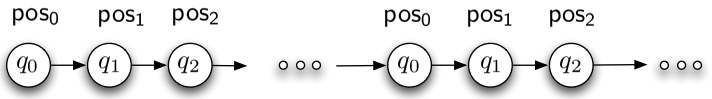
Exercise

Prove that the semantics does indeed match the definitions $\Diamond\varphi \equiv \top\,\mathcal{U}\,\varphi$ and $\Box\varphi \equiv \neg\Diamond\neg\varphi$.



$$\lambda, \pi \models \Diamond pos_1$$

$$\lambda' = \lambda[1, \infty], \pi \models pos_1$$
$$pos_1 \in \pi(\lambda'[0])$$



$$\lambda, \pi \models \Box\Diamond pos_1 \text{ iff}$$
$$\lambda[0, \infty], \pi \models \Diamond pos_1 \text{ and}$$
$$\lambda[1, \infty], \pi \models \Diamond pos_1 \text{ and}$$
$$\lambda[2, \infty], \pi \models \Diamond pos_1 \text{ and}$$
$$\ldots$$

Representation of paths

- Paths are *infinite entities*.
- They are theoretical constructs.
- We need a *finite representation*!
- We consider paths in a Kripke structure.

We use a *(pointed) Kripke model* $\mathfrak{M}, q$ and consider the problem whether an $\mathcal{L}_{LTL}$-formula holds on *all paths of* $\mathfrak{M}$ *starting in* q.

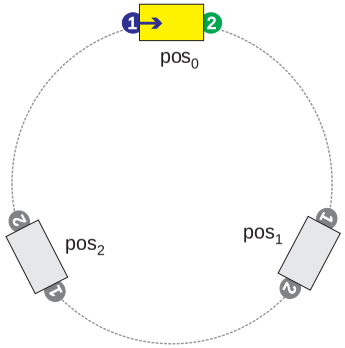
Definition 2.4 ($\mathfrak{M}$-path, $\Lambda_{\mathfrak{M}}(q)$)

An $\mathfrak{M}$*-path* (or *computation*) is given by $\lambda \in Q^{\omega}_{\mathfrak{M}}$ such that *subsequent states are connected by transitions* from $\mathcal{R}_{\mathfrak{M}}$. We use the same notation for these paths as introduced above. For $q \in Q$ we use $\Lambda_{\mathfrak{M}}(q)$ to denote the set of all $\mathfrak{M}$*-paths starting in* q and we define $\Lambda_{\mathfrak{M}}$ as $\bigcup_{q \in Q} \Lambda_{\mathfrak{M}}(q)$.

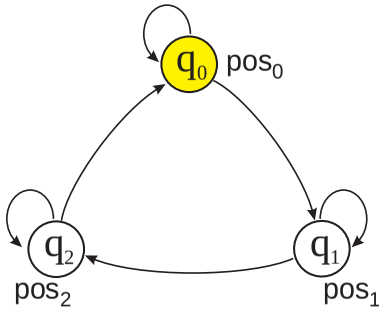
The subscript "$\mathfrak{M}$" is often omitted and we refer to an $\mathfrak{M}$-path simply as path when clear from context.

Computational vs. behavioural structure

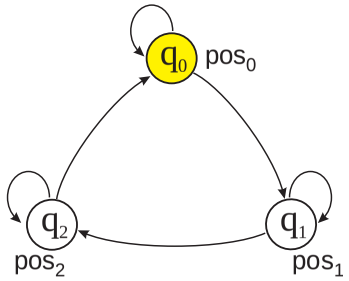
System



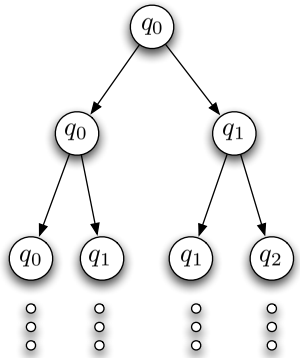
Computational str.



Computational str.



Behavioural str.



Important!

The *behavioural structure* is usually *infinite*! Here, it is an *infinite tree*. We say it is the q_0*-unraveling* of the model.

Some Exercises

Example 2.5

Formalise the following as **LTL** formulae:

1. r should never occur.
2. r should occur exactly once.
3. At least once r should directly be followed by s.

$\rightsquigarrow$ *blackboard*

2.2 Branching-Time Logics

Branching Time

- ***CTL, CTL****: *Computation Tree Logics*.
- Reasoning about possible *computations of a system*.
- Time is *branching*: We want all alternative computations included!
- *Models*: *states* (time points, situations), *transitions* (changes). ($\rightsquigarrow$ Kripke models).
- *Paths*: courses of action, computations. ($\rightsquigarrow$ **LTL**)

- *Path quantifiers*: A (for all paths), E (there is a path);
- *Temporal operators*: $\bigcirc$ (nexttime), $\Diamond$ (sometime), $\Box$ (always) and $\mathcal{U}$ (until);
- *CTL*: each *temporal operator* must be *immediately preceded by exactly one path quantifier*;
- *CTL**: no syntactic restrictions.

Definition 2.6 ($\mathcal{L}_{CTL^*}$ [Emerson and Halpern, 1986])

The *language* $\mathcal{L}_{CTL^*}(\mathcal{P}rop)$ is given by all formulae generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}\gamma$$

where

$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \gamma \mathcal{U} \gamma \mid \bigcirc\gamma$$

and $p \in \mathcal{P}rop$. Formulae φ (resp. γ) are called *state* (resp. *path*) formulae.

We use the same abbreviations as for $\mathcal{L}_{LTL}$.

- The $\mathcal{L}_{CTL^*}$-formula $\mathsf{E}\Diamond\varphi$, for instance, ensures that *there is at least one path* on which φ holds at some (future) time moment.
- The formula $\mathsf{A}\Diamond\Box\varphi$ states that φ holds *almost everywhere* . More precisely, on all paths it always holds from some future time moment.
- $\mathcal{L}_{CTL^*}$-formulae do not only talk about temporal patterns on a given path, *they also quantify* (existentially or universally) over such paths.
- The logic is complex! For practical purposes, a fragment with *better computational properties* is often sufficient.

Definition 2.7 ($\mathcal{L}_{CTL}$ [Clarke and Emerson, 1981])

The *language* $\mathcal{L}_{CTL}(\mathcal{P}rop)$ is given by all formulae generated by the following grammar, where $p \in \mathcal{P}rop$ is a proposition:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}(\varphi \mathcal{U} \varphi) \mid \mathsf{E}\bigcirc\varphi \mid \mathsf{E}\Box\varphi.$$

For example, $\mathsf{A}\Box\mathsf{E}\bigcirc p$ is a $\mathcal{L}_{CTL}$-formula whereas $\mathsf{A}\Box\Diamond p$ is not.

We define the *Boolean connectives* as usual.
It remains to define the other temporal operators. What about $\Diamond\varphi \equiv \neg\Box\neg\varphi$?

We introduce the following macros:

- $\Diamond\varphi \equiv \top \mathcal{U} \varphi$,
- $\mathsf{A}\bigcirc\varphi \equiv \neg\mathsf{E}\bigcirc\neg\varphi$,
- $\mathsf{A}\Box\varphi \equiv \neg\mathsf{E}\Diamond\neg\varphi$, and
- $\mathsf{A}\varphi \mathcal{U} \psi \equiv \ldots$ *Exercise!*

Example

Are the following **CTL*** or **CTL** formulae? What do the formulae express?

1. $\mathsf{E}\Diamond\mathsf{A}\bigcirc \mathit{shutdown}$
2. $\mathsf{E}\Diamond\bigcirc \mathit{shutdown}$
3. $\mathsf{A}\Box\Diamond \mathit{rain}$
4. $\mathsf{A}\Box\mathsf{A}\Diamond \mathit{rain}$ (Is it different from (3)?)
5. $\mathsf{E}\Diamond\Box \mathit{broken}$

The semantics is given over Kripke models with a *serial* transition relation (time flows forever!).

Definition 2.8 (Semantics $\models^{\mathsf{CTL}^*}$)

Let $\mathfrak{M}$ be a Kripke model, $q \in Q$ and $\lambda \in \Lambda$. The *semantics of $\mathcal{L}_{CTL^*}$- and $\mathcal{L}_{CTL}$-formulae* is given by the satisfaction relation $\models^{\mathsf{CTL}^*}$ for *state formulae* by

- $\mathfrak{M}, q \models^{\mathsf{CTL}^*} p$ iff $\lambda[0] \in \pi(p)$ and $p \in \mathcal{P}rop$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}^*} \neg\varphi$ iff $\mathfrak{M}, q \not\models^{\mathsf{CTL}^*} \varphi$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}^*} \varphi \wedge \psi$ iff $\mathfrak{M}, q \models^{\mathsf{CTL}^*} \varphi$ and $\mathfrak{M}, q \models^{\mathsf{CTL}^*} \psi$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}^*} \mathsf{E}\varphi$ iff *there is a path* $\lambda \in \Lambda(q)$ such that $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \varphi$;

and for *path formulae* by:

- $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \varphi$ iff $\mathfrak{M}, \lambda[0] \models^{\mathsf{CTL}^*} \varphi$;
- $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \neg\gamma$ iff $\mathfrak{M}, \lambda \not\models^{\mathsf{CTL}^*} \gamma$;
- $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \gamma \wedge \delta$ iff $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \gamma$ and $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \delta$;
- $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \bigcirc\gamma$ iff $\lambda[1,\infty], \pi \models^{\mathsf{CTL}^*} \gamma$; and
- $\mathfrak{M}, \lambda \models^{\mathsf{CTL}^*} \gamma\,\mathcal{U}\,\delta$ iff there is an $i \in \mathbb{N}_0$ such that $\mathfrak{M}, \lambda[i,\infty] \models^{\mathsf{CTL}^*} \delta$ and $\mathfrak{M}, \lambda[j,\infty] \models^{\mathsf{CTL}^*} \gamma$ for all $0 \leq j < i$.

Is this semantics over paths necessary for CTL?

State-based semantics for CTL

- $\mathfrak{M}, q \models^{\mathsf{CTL}} p$ iff $q \in \pi(p)$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}} \neg\varphi$ iff $\mathfrak{M}, q \not\models^{\mathsf{CTL}} \varphi$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}} \varphi \wedge \psi$ iff $\mathfrak{M}, q \models^{\mathsf{CTL}} \varphi$ and $\mathfrak{M}, q \models^{\mathsf{CTL}} \psi$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}} \mathsf{E}\bigcirc\varphi$ iff *there is a path* $\lambda \in \Lambda(q)$ such that $\mathfrak{M}, \lambda[1] \models^{\mathsf{CTL}} \varphi$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}} \mathsf{E}\Box\varphi$ iff *there is a path* $\lambda \in \Lambda(q)$ such that $\mathfrak{M}, \lambda[i] \models^{\mathsf{CTL}} \varphi$ for every $i \geq 0$;
- $\mathfrak{M}, q \models^{\mathsf{CTL}} \mathsf{E}\varphi\,\mathcal{U}\,\psi$ iff *there is a path* $\lambda \in \Lambda(q)$ such that $\mathfrak{M}, \lambda[i] \models^{\mathsf{CTL}} \psi$ for some $i \geq 0$, and $\mathfrak{M}, \lambda[j] \models^{\mathsf{CTL}} \varphi$ for all $0 \leq j < i$.

Example 2.9 (Robots and Carriage)

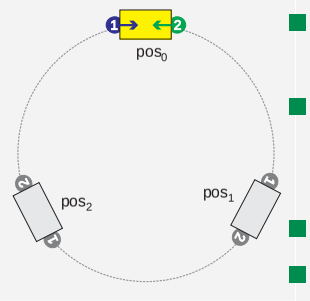
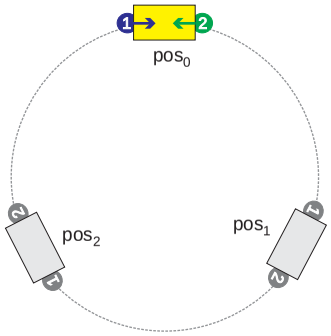


Figure 1: Two robots and a carriage.

- *Two robots* push a *carriage* from opposite sides.
- Carriage can *move clockwise or anticlockwise*, or it can remain in the same place.
- *3 positions* of the carriage.
- We label the states with *propositions* pos_0, pos_1, pos_2, respectively, to allow for referring to the current position of the carriage in the object language.



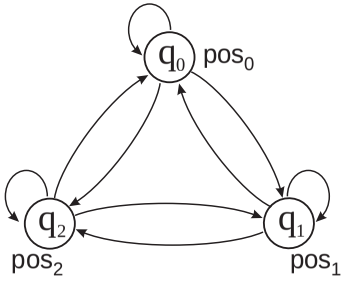
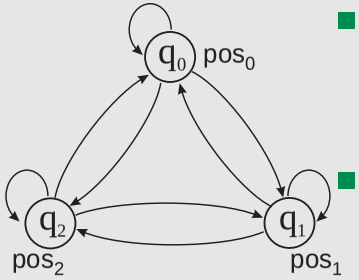


Figure 2: Two robots and a carriage: A schematic view (left) and a transition system $\mathfrak{M}_0$ that models the scenario (right).



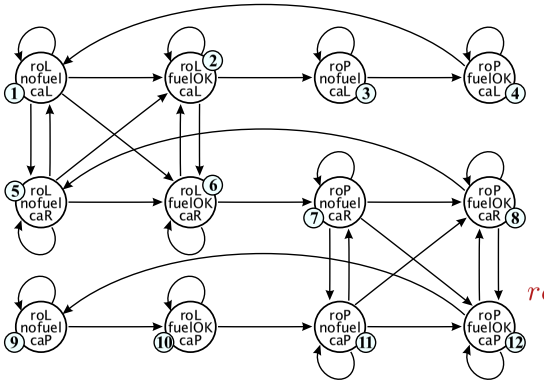
- $\mathfrak{M}_0, q_0 \models^{CTL} \mathsf{E}\Diamond pos_1$: In state q_0, there is a path such that the carriage will reach position 1 sometime in the future.
- The same is not true for *all* paths, so we also have: $\mathfrak{M}_0, q_0 \not\models^{CTL} \mathsf{A}\Diamond pos_1$.

It becomes more interesting if *abilities of agents are considered* $\rightsquigarrow$ ***ATL***.

Example: Rocket and Cargo

- A **rocket** and a **cargo**.
- The rocket can be moved between London (proposition roL) and Paris (proposition roP).
- The cargo can be in London (caL), Paris (caP), or inside the rocket (caR).
- The rocket can be moved only if it has its fuel tank full ($fuelOK$).
- When it moves, it consumes fuel, and $nofuel$ holds after each flight.

Example: Rocket and Cargo

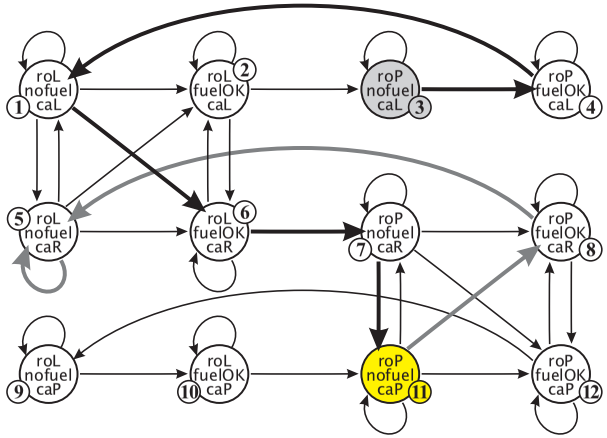


$roL \rightarrow \mathsf{E}\Diamond roP$

$\mathsf{A}\Box(roL \vee roP)$

$roL \rightarrow \mathsf{A}\bigcirc(roP \rightarrow nofuel)$

Example: Rocket and Cargo



$\mathsf{E}\Diamond caP$

2.3 References

R. Alur, T. A. Henzinger, and O. Kupferman (2002).
Alternating-time Temporal Logic.
Journal of the ACM, **49**:672–713.

E. A. Emerson (1990).
Temporal and modal logic.
Handbook of Theoretical Computer Science, volume B, 995–1072. Elsevier.

Fagin, R., Halpern, J. Y., Moses, Y. & Vardi, M. Y. (1995).
Reasoning about Knowledge.
MIT Press.

Wojtek Jamroga and Jürgen Dix (2008).
Model Checking Abilities of Agents: A Closer Look.
Theory of Computing Systems, **42(3)**, 366–410.

Wojtek Jamroga and Jürgen Dix (2005).
Do Agents Make Model Checking Explode (Computationally)?
M. Pechoucek and P. Petta and L.Z. Varga (Eds.), Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS '05), pages 398-407. LNCS 3690. Springer, 2005.

Kripke, S. (1963a).
Semantical analysis of modal logic I. Normal propositional calculi.
Zeitschrift für math. Logik und Grundlagen der Mathematik, 9, 67–96.

Kripke, S. (1963b).
Semantical considerations on modal logic.
Acta Philosophica Fennica, 16, 83–94.

3. Model Checking Temporal Logics

3.1 Motivation

Why do we need verification methods?

AT&T Telephone Network Outage (1990)

- Problem in New York City: *9 hour outage* of large parts of *US telephone network*.
- Costs: *several 100 million $*.
- Source: *wrong interpretation of a break statement in C*.

Acknowledgment: The following presentation is partly based on the book "Principles of Model Checking" by Christel Baier and Joost-Pieter Katoen.

Why do we need verification methods?

Pentium FDIV BUG (1994)

(FDIV: Floating point division unit)

- Incorrect results.
- Costs: *500 million $ and image loss*.
- Source: *flaw* in realisation of *floating-point division*.

Why do we need verification methods?

Ariane 5 Desaster (1996)

- Crash of Ariane 5-missle.
- Costs: *> 500 million $*.
- Source: *flaw in data conversion* from a 64-bit floating point to a 16-bit signed integer.

What are the lessons learned?

$\rightsquigarrow$ *Verification may pay off!*

In such cases the *extra costs* and efforts put into proper verification techniques may be *cheaper as the results of an error*.

Why do we need verification methods?

- Software becomes *larger*.
- Use in *safety-critical* systems, important domains.
- Increasing need for *reliable* software.

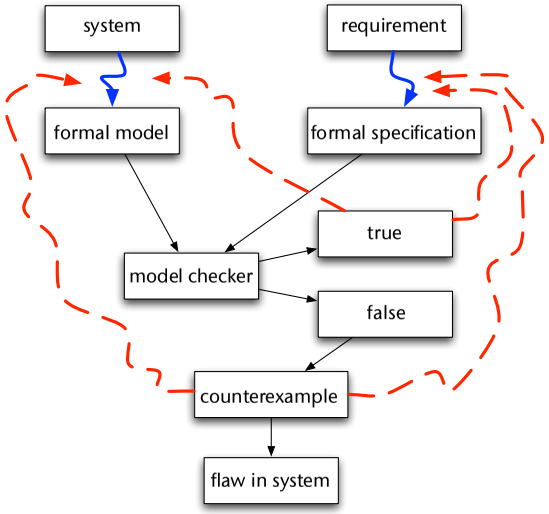
Why?

- Errors can be *costly and fatal* (Ariane-5 launch, stock market systems,...).
- *Mass production* of products (errors are expensive, computer chips,...).

Formal Verification Techniques

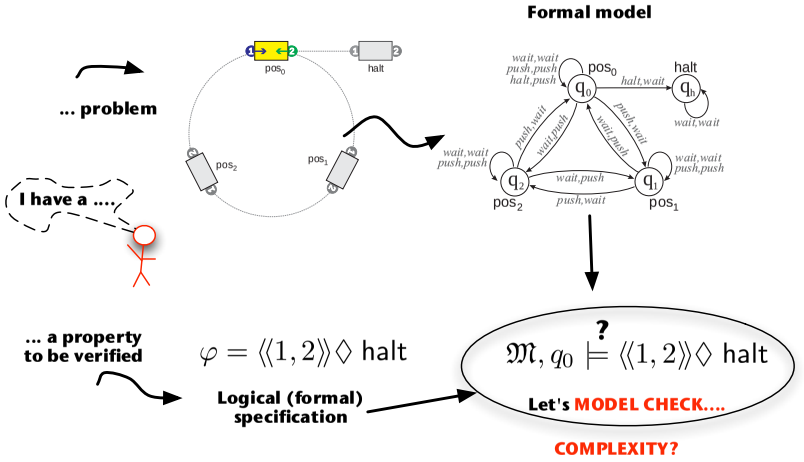
- *Testing and reviewing* ($\rightsquigarrow$ non-formal methods)
- *Deductive methods* (Hoare Calculus), code integration ($\rightsquigarrow$ *undecidable*, expertise during programming necessary)
- *Model checking* ($\rightsquigarrow$ how is the *correct model obtained*?)

What is Model Checking



3.2 The Model Checking Problem and CTL

What is Model Checking



What is Model Checking? (1)

- *Model checking* refers to the problem to determine whether a given formula φ *is satisfied in a state* q *of model* $\mathfrak{M}$.
- *Local model checking* is the decision problem that determines *membership* in the set $\mathsf{MC}(\mathcal{L}, \mathsf{Struc}, \models) := \{(\mathfrak{M}, q, \varphi) \in \mathsf{Struc} \times \mathcal{L} \mid \mathfrak{M}, q \models \varphi\}$, where
  - $\mathcal{L}$ is a *logical language*,
  - Struc is a *class of (pointed) models* for $\mathcal{L}$ (i.e. a tuple consisting of a model and a state), and
  - $\models$ is a *semantic satisfaction relation* compatible with $\mathcal{L}$ and Struc.

What is Model Checking? (2)

- It is often useful to compute the *set of states* in $\mathfrak{M}$ *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*.
- Here: The *complexities of local and global model checking coincide*.
- We are interested in the *decidability and the computational complexity* of determining whether an input instance $(\mathfrak{M}, q, \varphi)$ belongs to $\mathsf{MC}(\dots)$.

Input size

Important

The *complexity* is always relative to the *size of the input*!

- That is, the *size* of the *representation of the model* and the *representation of the formula* that we use.
- In order to establish the complexity, it is necessary to fix how we *represent* the input and how we *measure* its size.

Remark 3.1

Sometimes it makes sense to only consider the size of the model or of the formula.

In this course, we *always consider the size of the model and of the formula* .

Input size

- The *size of the model* ($|\mathfrak{M}|$) is given by the *number of transitions* in $\mathfrak{M}$, and the *size of the formula* ($|\varphi|$) is given by its *length* (i.e., the number of elements it is composed of, apart from parentheses).

Why do we not consider the number of states in a model?

For example, the formula $\mathsf{A}\bigcirc(pos_0 \vee pos_1)$ has length 5.

Be careful...

...if numbers are involved!

Model Checking LTL/CTL

Let $\mathfrak{M}$ be a *Kripke model* and q *be a state* in the model.

- *Model checking* a $\mathcal{L}_{CTL}/\mathcal{L}_{CTL^*}$-formula φ in $\mathfrak{M}, q$ means to determine whether $\mathfrak{M}, q \models \varphi$, i.e., whether φ holds in $\mathfrak{M}, q$.
- For LTL, checking $\mathfrak{M}, q \models \varphi$ means that we check the *validity* of φ in the pointed model $\mathfrak{M}, q$, i.e., whether φ holds *on all the paths* in $\mathfrak{M}$ that start from q.
- That is, it is *equivalent to* ***CTL**** *model checking of a formula* $\mathsf{A}\varphi$ in $\mathfrak{M}, q$.

Model Checking CTL

- Let the function $pre(Q')$ return *all states such that there is a transition leading to a state in* Q' .
- The following algorithm is based on the following *fixed-point characterisations*:

$$\mathsf{E}\Box\varphi \leftrightarrow \varphi \wedge \mathsf{E}\bigcirc\mathsf{E}\Box\varphi,$$
$$\mathsf{E}\varphi_1\,\mathcal{U}\,\varphi_2 \leftrightarrow \varphi_2 \vee (\varphi_1 \wedge \mathsf{E}\bigcirc\mathsf{E}\varphi_1\,\mathcal{U}\,\varphi_2).$$

Model Checking CTL

function $mcheck(\mathfrak{M}, \varphi)$.
case $\varphi \equiv p$: return $\{q \in Q \mid p \in \pi(q)\}$
 case $\varphi \equiv \neg\psi$: return $Q \setminus mcheck(\mathfrak{M}, \psi)$
 case $\varphi \equiv \psi_1 \wedge \psi_2$: return $mcheck(\mathfrak{M}, \psi_1) \cap mcheck(\mathfrak{M}, \psi_2)$
 case $\varphi \equiv \mathsf{E}\bigcirc\psi$: return $pre(mcheck(\mathfrak{M}, \psi))$
 case $\varphi \equiv \mathsf{E}\Box\psi$:
  $Q_1 := Q$; $Q_2 := Q_3 := mcheck(\mathfrak{M}, \psi)$;
  while $Q_1 \not\subseteq Q_2$ **do** $Q_1 := Q_1 \cap Q_2$; $Q_2 := pre(Q_1) \cap Q_3$ **od**;
  return Q_1
 case $\varphi \equiv \mathsf{E}\psi_1\,\mathcal{U}\,\psi_2$:
  $Q_1 := \emptyset$; $Q_2 := mcheck(\mathfrak{M}, \psi_2)$; $Q_3 := mcheck(\mathfrak{M}, \psi_1)$;
  while $Q_2 \not\subseteq Q_1$ **do** $Q_1 := Q_1 \cup Q_2$; $Q_2 := pre(Q_1) \cap Q_3$ **od**;
  return Q_1
 end case

Figure 3: **CTL**-model checking algorithm

Model Checking CTL

Theorem 3.2
(CTL [Clarke et al., 1986, Schnoebelen, 2003])

Model checking CTL *is* **P***-complete, and can be done in time* $\mathbf{O}(|\mathfrak{M}| \cdot |\varphi|)$, *where* $|\mathfrak{M}|$ *is given by the number of transitions.*

Proof

The algorithm determining the states in a model at which a given formula holds is presented in Figure 3 on Slide 123.
The *lower bound* (**P**-hardness) can be for instance proven by a *reduction of the Circuit-Value-Problem*
[Schnoebelen, 2003].

Model Checking LTL and CTL

Often, one is only interested in the *complexity class of model checking* and not in a specific algorithm and its detailed complexity. Is there a *more convenient way to determine the complexity* without working out the algorithm?

- *Automata-theory* to build algorithms.
- *Unified* approach.
- Automata are *well studied*.
- *Simplifies complexity* analysis.
- Usually, one is only interested in a *complexity class*. It is very time-demanding to come up with a *good* algorithm.

3.3 Büchi Automata

Büchi Automata

- We would like to use *finite automata* to solve the model checking problem.
- Finite automata (on finite words) accept only *finite words* but *paths are infinite*.
- We need to extend the model to *finite automata that accept infinite words*.

How can we accept infinite words?

Definition 3.3 (ω-automaton)

An *ω-automaton* is a tuple

$$A = (Q, \Sigma, \Delta, q_I, C)$$

where

1. Q is a finite set of *states*;
2. Σ is a *finite alphabet*;
3. $\Delta \subseteq Q \times \Sigma \times Q$ a *transition relation* ;
4. q_I is the *initial state*; and
5. C an *acceptance component* (which is specialised in the following).

The crucial point is the *acceptance component*!

Depending on the acceptance condition various types of automata arise (e.g., Büchi, Rabin, Muller automata).

Definition 3.4 (Run)

A *run* $\rho = \rho(0)\rho(1)\cdots \in Q^\omega$ *of* A *on a word* $w = w_1 w_2 \cdots \in \Sigma^\omega$ is an *infinite sequence* of states of A such that:

1. $\rho(0) = q_I$
2. $\rho(i) \in \Delta(\rho(i-1), w_i)$ for $i \geq 1$.

We define $Inf(\rho)$ as the set of *all states that occur infinitely often* on ρ; that is,

$$Inf(\rho) = \{q \in Q \mid \forall i \exists j (j > i \wedge \rho(j) = q)\}$$

Definition 3.5 (Büchi automaton)

A *Büchi automaton* is an ω*-automaton*

$$A = (Q, \Sigma, \Delta, q_I, F)$$

where $F \subseteq Q$ with the following *acceptance condition*: A *accepts* $w \in \Sigma^\omega$ if, and only if, *there is a run* ρ of A such that

$$Inf(\rho) \cap F \neq \emptyset.$$

Thus, such an automaton accepts all words such that *some state from* F *is visited infinitely often* on a corresponding run.

Definition 3.6 (Acceptable language)

The *language accepted by* A, $L(A)$, consists of all words accepted by A. That is,

$$L(A) = \{w \in \Sigma^\omega \mid A \text{ accepts } w\}.$$

A *language* is said to be *(Büchi) acceptable* if there is a Büchi automaton that *accepts* it.

Remark 3.7 (Other automata types)

Other acceptance conditions yield different automata types: Rabin automata, Muller automata.

Example 3.8

Is there a Büchi Automaton that accepts the following language L over $\Sigma = \{a, b, c\}$?

$$L = \{w \in \Sigma^\omega \mid w \text{ contains } \textit{infinitely} \text{ many } a \text{ or } b \text{ and only } \textit{finitely} \text{ many } c\}$$

$\rightsquigarrow$ *blackboard*

Example 3.9

Is there a Büchi Automaton that accepts the following language L over $\Sigma = \{a, b\}$?

$$L = \{w \in \Sigma^\omega \mid w \textit{ ends with } a^\omega \text{ or } (ab)^\omega\}$$

Proposition 3.10 (Closure propeties)

1. *Büchi acceptable languages are closed under union, intersection, and negation.*
2. *If A is a regular language with $\epsilon \notin A$, then, A^ω is Büchi acceptable.*
3. *If A is a regular language and B is Büchi recognizable, then AB is Büchi acceptable.*

Proof sketch

1. *Union*: Nondeterministically *guess which automata* should be executed.$\rightsquigarrow$ Exercise
   Intersection: *Product automaton* yields a generalised Büchi automaton. The *acceptance set* is given by $\{F_1 \times S_2, S_1 \times F_2\}$.$\rightsquigarrow$ Exercise
   Complement: This part is *non-trivial* and cannot be done in the scope of this lecture.
2. A^ω: *Connect transitions to final states* also *with the initial state* $\rightsquigarrow$ Exercise
3. AB: *Connect transitions to final states* of the finite automaton with the *initial state of the Büchi automaton*.$\rightsquigarrow$ Exercise

Theorem 3.11 (Characterization Theorem)

A language L is Büchi acceptable if, and only if, there are finitely many regular languages $U_1, \ldots, U_n$ and $V_1, \ldots, V_n$ such that

$$L = \bigcup_{i=1,\ldots,n} U_i (V_i)^\omega$$

This shows that *any language $L \neq \emptyset$* acceptable by a Büchi automaton contains an *ultimately periodic word*.

Example 3.12

For the language $L = \{w \in \Sigma^\omega \mid w \textit{ ends with } a^\omega \text{ or } (ab)^\omega\}$ from Example 3.9 we have that $L = \Sigma^*\{a\}^\omega \cup \Sigma^*\{ab\}^\omega$.

Proof of Theorem 3.11

"⇒": Let $W(q, q') = \{w \in \Sigma^* \mid q \rightarrow^w q'\}$. Each language $W(q, q')$ is *regular*. Then,

$$L(A) = \bigcup_{q \in Q_f} W(q_I, q)(W(q, q))^\omega.$$

"⇐": Let $L = \bigcup_{i=1,\ldots,n} U_i(V_i)^\omega$ where each U_i, V_i *is regular*. By Proposition 3.10 we have that $(V_i)^\omega$ and $U_i(V_i)^\omega$ are *Büchi recognizable*. Thus *also their finite union*.

Definition 3.13 (Generalised Büchi automaton)

A *generaliaed Büchi automaton* is an ω-automaton

$$A = (Q, \Sigma, \Delta, q_I, F)$$

where $F \subseteq \mathcal{P}(Q)$ with the following *acceptance condition*: A *accepts* $w \in \Sigma^\omega$ if, and only if, there is a run ρ of A such that *for each* $F_i \in F$

$$Inf(\rho) \cap F_i \neq \emptyset.$$

Thus, such an automaton *accepts* all words such that *some state* from *each* F_i is *visited infinitely often* on a corresponding run.

We will use generalised Büchi automata for model checking **LTL**. *How is the relation between Büchi and generalised Büchi automata?*

Proposition 3.14 (Generalised Büchi ⇝ Büchi)

For each generalised Büchi automaton one can construct an equivalent Büchi automaton.

Proof.

Idea: Consider *state-tuples:* $S \times \{1, \ldots, k\}$. If the GBA moves to the next acceptance set a *counter* is incremented (modulo k). Then, a run visits states from *each* F_i infinitely often iff *states from* $F_1 \times \{1\}$ appear infinitely often. □

Proof ctd.

Let $A = (\Sigma, S, \Delta, S_0, \{F_1, \ldots, F_n\})$ be a generalised Büchi automaton. We construct the Büchi Automaton $A' = (\Sigma, S', \Delta', S'_0, F')$:

- $S' = S \times \{1, \ldots, n\}$;
- $S'_0 = S_0 \times \{1\}$;
- $((s, j), a, (t, i)) \in \Delta'$ iff

$$(s, a, t) \in \Delta \text{ and } \begin{cases} i = j & \text{, if } s \notin F_j; \\ i = (j + 1) \mod k & \text{, if } s \in F_j; \end{cases}$$

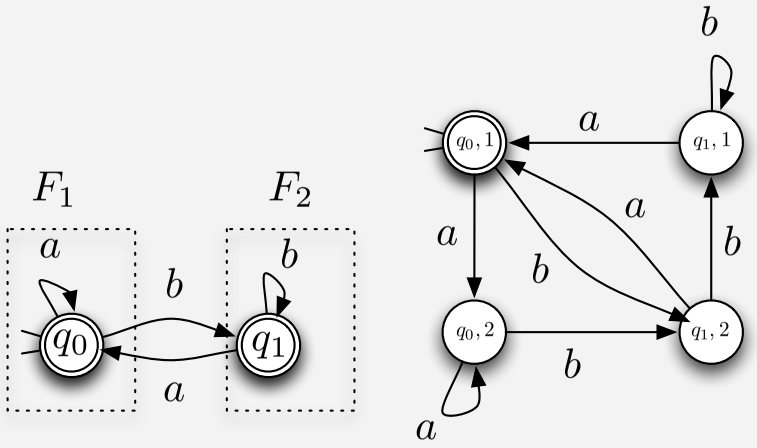
- $F' = F_1 \times \{1\}$.

Proof ctd.

It remains to prove that *both automata accept the same languages*. We present the main ideas.
"⇒": Let A be a GBA that accepts the word w. Then, there is a *run* ρ such that *states from each* F_i, $i = 1, \ldots, k$, *occur infinitely often* on ρ. That is, there is an *infinite subsequence* $(q_1 \ldots q_k)^\omega$ of ρ such that $q_i \in F_i$. Hence, the *state* $(q_1, 1)$ *is visited infinitely often* in the automaton A'.

"⇐": Let A' accept the word w. Then, some state $(q_1, 1)$ with $q_1 \in F_1$ is *visited infinitely often*. After it has been visited once the automaton is in a state $(q, 2)$ and can *only return to* $(q', 1)$ if some state $q \in F_2$ *is visited*, some from F_3 and so on is visited.

Example 3.15



Checking Emptiness

For the model checking algorithms we need to check whether the language of a Büchi automaton is empty.

Definition 3.16 (Graph reachability)

Let $G = (V, E)$ be graph. Given two vertices $u, v \in V$ the *graph-reachability problem* is the question whether v is reachable from u.

Theorem 3.17 ([Jones, 1977, Jones, 1975])

The graph-reachability problem is $NLOGSPACE$*-complete under logspace-reductions.*

Theorem 3.18 ([Emerson and Lei, 1987])

The emptiness problem for Büchi automata is solvable in linear time and in nondeterministic logarithmic space .

Proof

We check whether there is some *ultimately periodic word* by *finding an accepting* state *reachable from the initial* state *and from itself*. The following algorithm runs in non-deterministic logarithmic space:

1. *Guess* an accepting state r, and
2. *check* whether $reach(r, r)$.

How does $reach(x, y)$ work?

1. Chose some x-successor x' (*non-determinism!*).
2. Return "yes", if $x' = y$ else $reach(x', y)$.

Hardness is shown by a reduction of the $NLOGSPACE$-complete problem of *graph reachability* from Definition 3.16. Given G, u, v, *transform G to a Büchi automaton* with *initial state u* and *final state v* and add a *loop to v*. Then:

v reachable from u in G iff automaton non-empty.

□

3.4 Model Checking LTL

Automata and Model Checking

How can we use *automata* for the model checking problem?

The basic idea is the following:

1. We build an automaton $A_{\mathfrak{M},q_0}$ *accepting the paths of model* $\mathfrak{M}, q_0$.
2. We build an automaton A_φ *accepting all paths satisfying* φ.
3. Then, we have:

$$\mathfrak{M} \models \varphi \text{ iff } L(A_{\mathfrak{M},q_0}) \subseteq L(A_\varphi).$$

Büchi Automata and Kripke Models

We can relate a *Kripke model* $\mathfrak{M} = (Q, \mathcal{R}, \pi)$ and a state $q_0 \in Q$ to a *Büchi automaton* $A_{\mathfrak{M},q_0} = (\Sigma, Q, q_0, \Delta, Q)$ where

- $\Sigma = \mathcal{P}(\mathcal{P}rop)$: Each input symbol is a *set of propositions*,
- $q' \in \Delta(q, w)$ iff $((q, q') \in \mathcal{R}$ *and* $w = \pi(q))$,
- *all states being accepting states* (i.e. each infinite run of the automaton is accepting).

Important

The automaton accepts words over $\mathcal{P}(\mathcal{P}rop)$ but paths are sequences of *states*! What now?

LTL Semantics Revisited

The truth of $\lambda, \pi \models \varphi$ does *only* depend on the *propositions* true at states.

Clearly, for path λ, λ' we have the following:

If *for all* $i \in \mathbb{N}_0$

$$\pi(\lambda[i]) = \pi(\lambda'[i]) \text{ then } \lambda, \pi \models \varphi \textit{ iff } \lambda', \pi \models \varphi.$$

Hence, we can also use the *infinite word*

$$\lambda^\pi := \pi(\lambda[0])\pi(\lambda[1])\pi(\lambda[2]) \cdots \in \mathcal{P}(\mathcal{P}rop)^\omega$$

to give *truth to* LTL*-formulae*.

How do the semantic clauses change?

Alternative LTL Semantics

The *original clauses* had the following form:

- $\lambda, \pi \models^{LTL} p$ iff $\lambda[0] \in \pi(p)$;
- $\lambda, \pi \models^{LTL} \neg\varphi$ iff $\lambda, \pi \not\models^{LTL} \varphi$;
- $\lambda, \pi \models^{LTL} \varphi \wedge \psi$ iff $\lambda, \pi \models^{LTL} \varphi$ and $\lambda, \pi \models^{LTL} \psi$.

What happens if we use λ^π instead of λ, π?

We simply *replace "λ, π" by "λ^π"* everywhere and modify the *clause for propositions* as follows:

- $\lambda^\pi \models^{LTL} p$ iff $p \in \lambda^\pi[0]$.

Note, we use the same notations for λ^π as for paths!

We can state the relation between $\Lambda_{\mathfrak{M}}$, $\mathfrak{M}$, q and $A_{\mathfrak{M},q}$ precisely.

Proposition 3.19

Let $\mathfrak{M} = (Q, \mathcal{R}, \pi)$ and $q_0 \in Q$. The automaton $A_{\mathfrak{M},q_0}$ accepts the language

$$\{\lambda^\pi \mid \lambda \in \Lambda_{\mathfrak{M}}(q_0)\}.$$

Proof.

Exercise! □

In the following we define the *automaton A_φ accepting* exactly those *infinite words w* over $\mathcal{P}(\mathcal{P}rop)$ such that $w \models \varphi$. Then, we have:

$$\mathfrak{M}, q \models \varphi \text{ iff } L(A_{\mathfrak{M},q}) \subseteq L(A_\varphi) \text{ iff } L(A_{\mathfrak{M},q}) \cap \overline{L(A_\varphi)} = \emptyset.$$

How can we *avoid the complementation* of the Büchi automaton (this operation is expensive)? We have:

$$L(A_{\mathfrak{M},q}) \cap \overline{L(A_\varphi)} = \emptyset \text{ iff } L(A_{\mathfrak{M},q}) \cap L(A_{\neg\varphi}) = \emptyset.$$

The Automaton A_φ

In the following we are concerned with construction the automaton A_φ.

Theorem 3.20 ([Sistla and Clarke, 1985, Lichtenstein and Pnueli, 1985, Vardi and Wolper, 1986])

For a given $\mathcal{L}_{LTL}$-formula φ a Büchi Automaton $A_\varphi = (S, \Sigma, \Delta, S_0, F)$ accepting exactly the words satisfying φ can be constructed where $\Sigma = \mathcal{P}(\mathcal{P}rop)$ and $|S| \leq 2^{(\mathcal{O}(|\varphi|))}$.

In the following we introduce additional notation and construct the automaton.

How does the automaton look like?

- *States* will consist of *subformulae of φ* (or their negations).
- A *run $\rho = S_1 S_2 \ldots$* of the automaton is an *infinite sequence of such subformulae* sets.

Note that we showed how to interpret LTL formulae over words of $\mathcal{P}(\mathcal{P}rop)$.

Given such a word $\lambda^\pi = w_1 w_2 \ldots$ we would like to *extend each w_i* with *subformulae to S_i* such that

$$\lambda^\pi[i, \infty] \models \psi \quad \textit{iff} \quad \psi \in S_i.$$

The basic idea is that a *run models the **LTL** semantics*.

Definition 3.21 (Closure $cl(\varphi)$)

The *closure $cl(\varphi)$* is defined as follows:

1. $\varphi \in cl(\varphi)$,
2. $\phi \wedge \psi \in cl(\varphi)$ implies $\phi, \psi \in cl(\varphi)$,
3. $\neg\psi \in cl(\varphi)$ implies $\psi \in cl(\varphi)$,
4. $\psi \in cl(\varphi)$ and $\psi \neq \neg\phi$ implies $\neg\psi \in cl(\varphi)$,
5. $\bigcirc\psi \in cl(\varphi)$ implies $\psi \in cl(\varphi)$,
6. $\psi \mathcal{U} \phi \in cl(\varphi)$ implies $\psi, \phi \in cl(\varphi)$.

Note, that it holds that $|cl(\varphi)| \leq 2|\varphi|$.

Example 3.22 (Closure)

How does the closure for $\varphi = r\,\mathcal{U}\,(s \vee t)$ look like?
The closure $cl(\varphi)$ consists of the following formulae:

1. φ
2. $s \vee t$
3. r
4. s
5. t

and their negations!

What other properties should such sets fulfill? Note, that we are interested in a *correspondence to runs*.

Definition 3.23 (Logically consistent)

We call $B \subseteq cl(\varphi)$ *logically consistent* iff for all $\varphi_1 \wedge \varphi_2, \psi \in cl(\varphi)$:

1. $\varphi_1 \wedge \varphi_2 \in B$ iff $\varphi_1 \in B$ *and* $\varphi_2 \in B$,
2. $\psi \in B$ implies $\neg\psi \notin B$,
3. $\top \in cl(\varphi)$ implies $\top \in B$.

We identify $\neg\neg\varphi$ with φ.

Definition 3.24 (Locally consistent)

We call $B \subseteq cl(\varphi)$ *locally consistent* iff for all $\varphi_1 \mathcal{U} \varphi_2 \in cl(\varphi)$:

1. $\varphi_2 \in B$ implies $\varphi_1 \mathcal{U} \varphi_2 \in B$.
2. $\varphi_1 \mathcal{U} \varphi_2 \in B$ and $\varphi_2 \notin B$ implies $\varphi_1 \in B$.

Definition 3.25 (Maximal consistent)

We call $B \subseteq cl(\varphi)$ *maximal* iff for all $\psi \in cl(\varphi)$

$$\psi \notin B \quad \text{implies} \quad \neg\psi \in B.$$

We identify $\neg\neg\varphi$ with φ.

Definition 3.26 (Elementary, $\mathcal{EL}(\varphi)$)

We call $B \subseteq cl(\varphi)$ *elementary* iff B is *logically* and *locally consistent* and *maximal*.
We define $\mathcal{EL}(\varphi)$ as the *set of all elementary subsets* of $cl(\varphi)$.

In the following we construct infinite words over $\mathcal{EL}(\varphi)$ that corresponds to accepting paths.

The closure of $\varphi = r \mathcal{U} s$ is given by $\{\varphi, \neg\varphi, r, s, \neg r, \neg s\}$.
Which of the following sets are *elementary*?

1. $\emptyset$
2. $\{r \mathcal{U} s, r, s\}$
3. $\{r \mathcal{U} s, r\}$
4. $\{r \mathcal{U} s, \neg r, \neg s\}$
5. $\{r \mathcal{U} s, \neg r, s\}$
6. $\{r \mathcal{U} s, r, \neg s\}$
7. $\{r \mathcal{U} s, r, \neg r, \neg s\}$
8. $\{\neg r \mathcal{U} s, r, \neg s\}$
9. $\{\neg r \mathcal{U} s, \neg r, \neg s\}$

Example 3.27 (Elementary sets)

The closure of $\varphi = r \mathcal{U} s$ is given by

$$cl(\varphi) = \{\varphi, \neg\varphi, r, s, \neg r, \neg s\}$$

The following list contains all *elementary sets* of φ:

1. $E_1 = \{r \mathcal{U} s, r, s\}$
2. $E_2 = \{r \mathcal{U} s, \neg r, s\}$
3. $E_3 = \{r \mathcal{U} s, r, \neg s\}$
4. $E_4 = \{\neg r \mathcal{U} s, r, \neg s\}$
5. $E_5 = \{\neg r \mathcal{U} s, \neg r, \neg s\}$

It is easily seen that we have the following *fixed-point equivalence*

$$\varphi_1 \mathcal{U} \varphi_2 = \varphi_2 \vee (\varphi_1 \wedge \bigcirc \varphi_1 \mathcal{U} \varphi_2).$$

We construct a path over $\mathcal{EL}(\varphi)$:

Definition 3.28 (φ-closure-labelling)

A *φ-closure-labelling* is a function

$$\tau : \mathbb{N}_0 \to \mathcal{EL}(\varphi)$$

such that:

(C1) $\bigcirc \varphi \in \tau(i)$ iff $\varphi \in \tau(i+1)$,

(C2) $\varphi_1 \mathcal{U} \varphi_2 \in \tau(i)$ iff
$\varphi_2 \in \tau(i)$ *or* $(\varphi_1 \in \tau(i)$ *and* $\varphi_1 \mathcal{U} \varphi_2 \in \tau(i+1))$,

(C3) $\varphi_1 \mathcal{U} \varphi_2 \in \tau(i)$ implies $\exists j (j \geq i$ *and* $\varphi_2 \in \tau(j))$.

(C1) – (C3) mirror the semantics of *path formulae of* LTL.

Remark 3.29

The fixed-point equivalence is modelled by (C2). Still, the "valid" closure labelling has to ensure that sometimes φ_2 becomes eventually the case. This is captured by (C3).

Given a word λ^π a *closure labelling corresponding* to λ^π should *agree with the propositional symbols*.

Definition 3.30 (λ^π-valid)

A *φ-closure-labelling τ* is said to be *λ^π-valid* iff *for all $p \in \mathcal{P}rop$* it holds that

1. $p \in \tau(i)$ implies $p \in \lambda^\pi[i]$, and
2. $\neg p \in \tau(i)$ implies $p \notin \lambda^\pi[i]$.

Lemma 3.31 (Soundness Lemma)

Let $\varphi \in \mathcal{L}_{LTL}(\mathcal{P}rop)$ and τ be a λ^π-valid closure labelling. Then, for all $\varphi' \in cl(\varphi)$ and all $i \geq 0$ it holds that

$$\varphi' \in \tau(i) \quad \textit{iff} \quad \lambda^\pi[i, \infty] \models \varphi'.$$

Lemma 3.32 (Existence Lemma)

Let $\varphi \in \mathcal{L}_{LTL}(\mathcal{P}rop)$. If $\lambda^\pi \models \varphi$. Then, there is a λ^π-valid φ-closure labelling τ such that $\varphi \in \tau(0)$.

Proof of Lemma 3.31.

The proof is done by structural induction on φ'.
$\rightsquigarrow$ Exercise! □

Proof of Lemma 3.32.

The *labelling is constructed from subformulae true at each point of λ^π.*
$\rightsquigarrow$ Exercise! □

From these lemmata we obtain the following theorem.

Theorem 3.33

Let $\varphi \in \mathcal{L}_{LTL}(\mathcal{P}rop)$. Then, $\lambda^\pi \models \varphi$ iff there is a λ^π-valid φ-closure labelling τ such that $\varphi \in \tau(0)$.

Now we proceed with the proof of *Theorem 3.20*.

Proof Idea

$$\lambda = q_0 q_1 q_2 \ldots$$

$\lambda, \pi \models \varphi$ iff $\lambda^\pi \models \varphi$

$$\lambda^\pi = \pi(q_0)\pi(q_1)\pi(q_2)\ldots$$

$\lambda^\pi \models \varphi$ iff
τ is λ^π-valid φ-closure labelling iff
τ accepted by the automaton

$$\tau = B_0 B_1 B_2 \ldots$$

run of the automaton given λ^π

Proof of Theorem 3.20

Using Theorem 3.33 we build a *generalised Büchi automaton accepting* all the *infinite words* λ^π that *correspond to a* λ^π*-valid* φ*-closure-labelling*.

Idea:

1. The automaton *reads* λ^π.
2. Each symbol causes a state change, *states are elementary sets*.
3. *Runs* ρ of the automaton *correspond to* φ*-closure labellings*.
4. ρ is *accepted* iff it is λ^π*-valid* and *satisfies* φ.

The automaton is defined as $A = (\Sigma, S, \Delta, S_0, F)$ where

1. $\Sigma = \mathcal{P}(\mathcal{P}rop)$
2. $S = \mathcal{EL}(\varphi)$
3. $S_0 = \{s \in S \mid \varphi \in s\}$
4. $F \rightsquigarrow$ see below
5. $(s, a, t) \in \Delta$ iff
   1. $s \cap \mathcal{P}rop = a$
   2. $\forall \bigcirc\psi \in cl(\varphi) : \bigcirc\psi \in s$ *iff* $\psi \in t$
   3. $\forall \varphi_1 \mathcal{U} \varphi_2 \in cl(\varphi) :$
      $\varphi_1 \mathcal{U} \varphi_2 \in s$ *iff* $(\varphi_2 \in s$ *or* $(\varphi_1 \in s$ *and* $\varphi_1 \mathcal{U} \varphi_2 \in t))$

Proof of Theorem 3.20 ctd.

How to define the set of accepting states?
We need to ensure that condition (C3) of a φ-closure-labelling is satisfied; that is, that *eventualities become actually satisfied*.

So, once a state containing an eventuality $\varphi_1 \mathcal{U} \varphi_2$ is visited sometime in the future a state containing φ_2 must be visited.

We require that *states containing*

$$(\varphi_2 \text{ and } \varphi_1 \mathcal{U} \varphi_2) \text{ or not } \varphi_1 \mathcal{U} \varphi_2$$

occur *infinitely often*. □

Proof of Theorem 3.20 ctd.

So, let $\varphi_1 \mathcal{U} \psi_1, \ldots, \varphi_n \mathcal{U} \psi_n$ be all eventualities occuring in $cl(\varphi)$. Then, we define $F = \{F_1, \ldots, F_n\}$ with

$$F_i = \{s \in S \mid \{\varphi_i \mathcal{U} \psi_i, \psi_i\} \subseteq s \text{ or } \varphi_i \mathcal{U} \psi_i \notin s\}.$$

That is,

$$F = \{\{s \in Q \mid \varphi_1 \mathcal{U} \varphi_2 \notin s \text{ or } \varphi_2 \in s\} \mid \varphi_1 \mathcal{U} \varphi_2 \in cl(\varphi)\}.$$

□

Proof of Theorem 3.20 ctd.

In line with Theorem 3.33 we have to show that *A accepts λ^π iff there is an accepting run ρ with $\varphi \in \rho(0)$ and which is an λ^π-valid φ-closure labelling*. This is immediate by construction.

Finally, we *convert the generalised Büchi automaton to a Büchi automaton* (cf. Proposition 3.14).
The number of states of the automaton is exponential in the length of the formula. □

Encoding as Generalised Büchi Automaton

How did we encode the logical elements?

- Semantics of *propositional logic*? $\rightsquigarrow$ *states*
- $\bigcirc$-operator? $\rightsquigarrow$ *transition relation*
- $\mathcal{U}$-operator? $\rightsquigarrow$ *states* plus *transition relation* plus *acceptance condition*

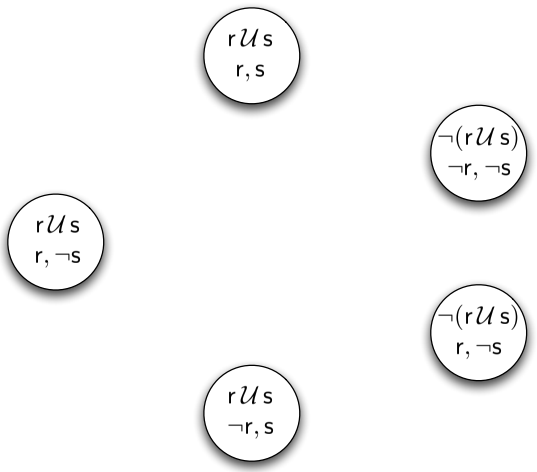
$$\varphi_1 \mathcal{U} \varphi_2 = \varphi_2 \vee (\varphi_1 \wedge \bigcirc \varphi_1 \mathcal{U} \varphi_2)$$

Example 3.34

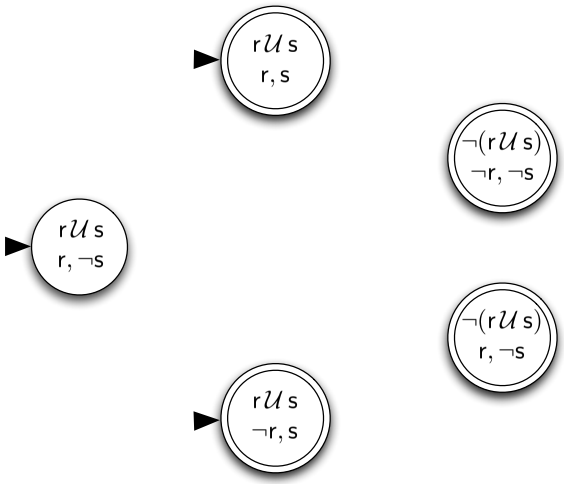
We consider the formula $\varphi = r \mathcal{U} s$. *Elementary sets* of φ:

1. $E_1 = \{r \mathcal{U} s, r, s\}$
2. $E_2 = \{r \mathcal{U} s, \neg r, s\}$
3. $E_3 = \{r \mathcal{U} s, r, \neg s\}$
4. $E_4 = \{\neg r \mathcal{U} s, r, \neg s\}$
5. $E_5 = \{\neg r \mathcal{U} s, \neg r, \neg s\}$

Constructing the Automaton for $r \mathcal{U} s$

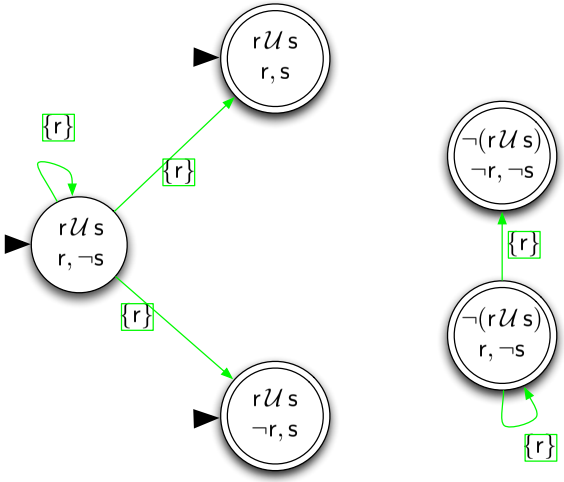


- Initial states?
  $\{s \in S \mid \varphi \in s\}$
- Accepting states?
  If $\varphi_1 \mathcal{U} \varphi_2 \in cl(\varphi)$ then
  $\varphi_1 \mathcal{U} \varphi_2 \notin s$ or
  $\varphi_2 \in s$



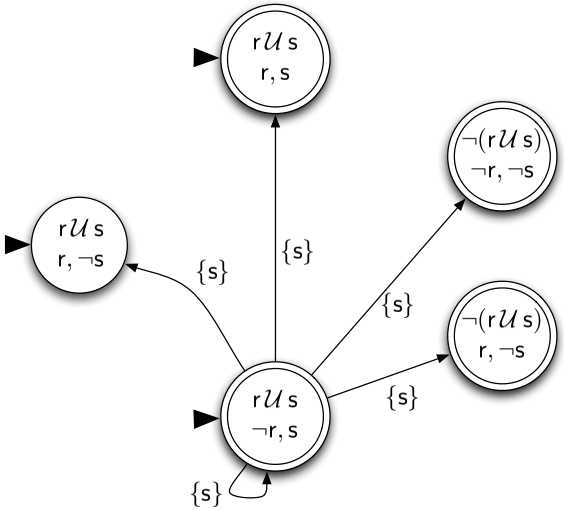
- Initial states?
  $\{s \in S \mid \varphi \in s\}$
- Accepting states?
  If $\varphi_1 \mathcal{U} \varphi_2 \in cl(\varphi)$ then
  $\varphi_1 \mathcal{U} \varphi_2 \notin s$ or
  $\varphi_2 \in s$
- $\rightsquigarrow$ *A reads* $\{r\}$

$\forall r \mathcal{U} s \in cl(\varphi) : r \mathcal{U} s \in s$ *iff* $(s \in s$ *or* $(r \in s$ *and* $r \mathcal{U} s \in t))$



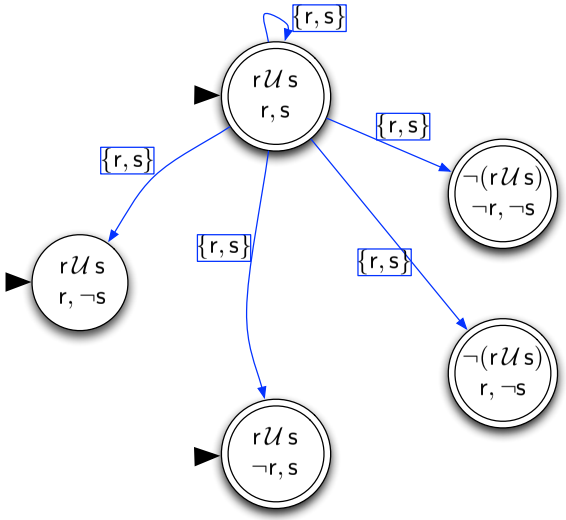
- *A reads* $\{r\}$
- $\rightsquigarrow$ *A reads* $\{s\}$

$\forall r \mathcal{U} s \in cl(\varphi) : r \mathcal{U} s \in s$ *iff* $(s \in s$ *or* $(r \in s$ *and* $r \mathcal{U} s \in t))$



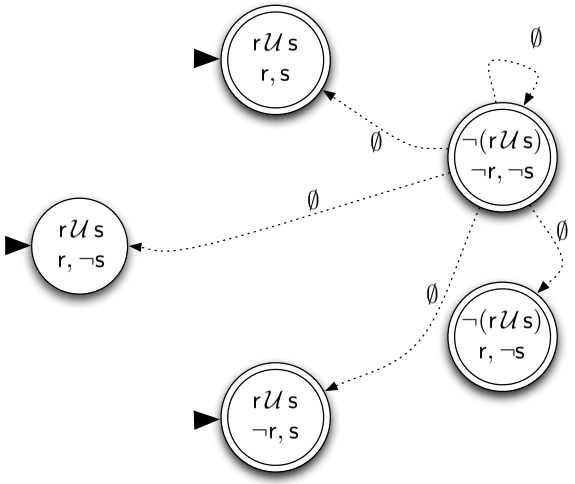
- *A reads* $\{s\}$
- $\rightsquigarrow$ *A reads* $\{r, s\}$

$\forall r \mathcal{U} s \in cl(\varphi) : r \mathcal{U} s \in s$ *iff* $(s \in s$ *or* $(r \in s$ *and* $r \mathcal{U} s \in t))$



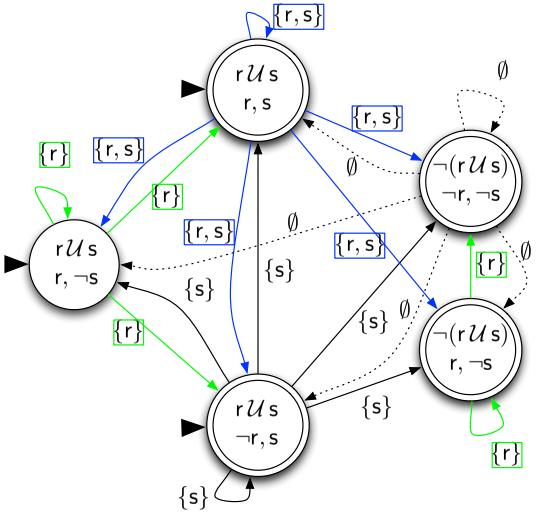
- A *reads* $\{r, s\}$
- $\rightsquigarrow$ A *reads* $\emptyset$

$\forall r \mathcal{U} s \in cl(\varphi) : r \mathcal{U} s \in s$ *iff* $(s \in s$ *or* $(r \in s$ *and* $r \mathcal{U} s \in t))$



- A *reads* $\emptyset$

$\forall r \mathcal{U} s \in cl(\varphi) : r \mathcal{U} s \in s$ *iff* $(s \in s$ *or* $(r \in s$ *and* $r \mathcal{U} s \in t))$



- The complete automaton

$\forall r \mathcal{U} s \in cl(\varphi) : r \mathcal{U} s \in s$ *iff* $(s \in s$ *or* $(r \in s$ *and* $r \mathcal{U} s \in t))$

Theorem 3.35 (LTL [Sistla and Clarke, 1985, Lichtenstein and Pnueli, 1985, Vardi and Wolper, 1986])

Model checking ***LTL*** *is* $PSPACE$*-complete, and can be done in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|\mathfrak{M}|)$*, where* $|\mathfrak{M}|$ *is given by the number of transitions.*

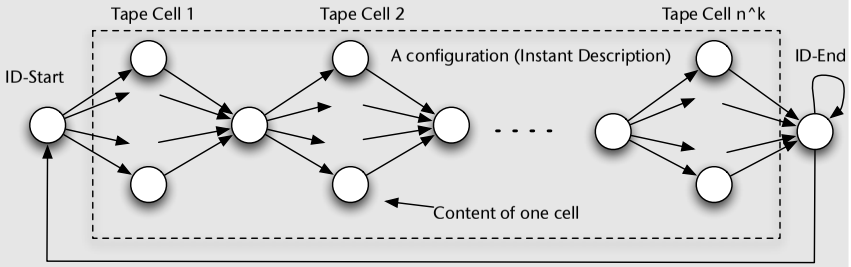
Proof: Upper Bound

Given an $\mathcal{L}_{LTL}$-formula φ.

1. *Construct Büchi automaton* $\mathcal{A}_{\neg\varphi}$ of size $2^{\mathbf{O}(|\varphi|)}$ accepting exactly the *words satisfying* $\neg\varphi$.
2. *Kripke model* $\mathfrak{M}, q$ can directly be interpreted as a Büchi automaton $\mathcal{A}_{\mathfrak{M},q}$ of size $\mathbf{O}(|\mathfrak{M}|)$ accepting all *possible words in the Kripke model* starting in q.
3. The model checking problem reduces to the *non-emptiness check of* $L(\mathcal{A}_{\mathfrak{M},q}) \cap L(\mathcal{A}_{\neg\varphi})$ which can be done in time $\mathbf{O}(|\mathfrak{M}|) \cdot 2^{\mathbf{O}(|\varphi|)}$ by constructing the product automaton.

Proof: Lower Bound

Simulate n^k*-space bounded deterministic Turing machine* $A = (S, \Sigma, \delta, s_0, S_f)$.



$\mathcal{P}rop = (S \times \Sigma) \cup \Sigma \cup \{ID - Start, ID - End\}$

Proof: Lower Bound

A *path* will be related to a sequence of *instantaneous descriptions*.

1. Use n^k $\bigcirc$-operators to *describe an ID*.
2. ψ_w: Encodes the *input* w.
3. ψ_{valid}: Checks whether an *ID is valid*.
4. ψ_{next}: Ensures that each *successive ID* follows from the current one.
5. ψ_{accept}: Describes the *halting configurations*.

Let $\psi := \psi_w \wedge \psi_{\text{valid}} \wedge \psi_{\text{next}} \wedge \psi_{\text{accept}}$. Then, we have

$\mathfrak{M}, q_0 \not\models \neg\psi$ iff $\exists \lambda \in \Lambda(q_0) : \lambda, \pi \models \psi$ iff A *accepts* w.

3.5 Model Checking CTL*

Theorem 3.36
(CTL* [Clarke et al., 1986, Emerson and Lei, 1987])

*Model checking **CTL*** *is $PSPACE$-complete, and can be done in time $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|\mathfrak{M}|)$, where $|\mathfrak{M}|$ is given by the number of transitions.*

Proof.

The *hardness* of **CTL*** model checking is immediate from Theorem 3.35 as $\mathcal{L}_{LTL}$ "can be seen" as a fragment of $\mathcal{L}_{CTL^*}$. □

Upper bound: *Combine* **CTL** and **LTL** model checking.

- Consider $\mathcal{L}_{CTL^*}$-formula φ containing $\mathsf{E}\psi$ *where ψ is a pure $\mathcal{L}_{LTL}$-formula*.
- *Determine all states which satisfy* $\mathsf{E}\psi$ (these are all states q with $\mathfrak{M}, q \not\models^{\mathbf{LTL}} \neg\psi$), Complexity: $PSPACE$.
- *Label them by a fresh proposition*, say p, and replace $\mathsf{E}\psi$ in φ by p: $\mathsf{E}\bigcirc(\overbrace{r \wedge \underbrace{\mathsf{E}\Diamond s}_{p_1}}^{p_2}) \rightsquigarrow \mathsf{E}\bigcirc(p_2 \wedge p_1)$

Applying this procedure *recursively* yields a pure $\mathcal{L}_{CTL}$-formula which can be verified in polynomial time.
Complexity: $\mathbf{P}^{PSPACE} = PSPACE$ □

This is a standard approach often used!

Summary

- Model checking ***CTL*** *is $\mathbf{P}$-complete*.
- Model checking ***LTL*** *is $PSPACE$-complete*. The algorithm has been constructed from Büchi automata.
- Model checking ***CTL**** *is also $PSPACE$-complete*. The algorithm is obtained by the ones for **CTL** and **LTL**.

3.6 References

Alur, R., Henzinger, T. A., and Kupferman, O. (1997).
Alternating-time Temporal Logic.
In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press.

Alur, R., Henzinger, T. A., and Kupferman, O. (2002).
Alternating-time Temporal Logic.
Journal of the ACM, 49:672–713.

Baier, C. and Katoen, J.-P. (2008).
Principles of Model Checking.
The MIT Press.

Beeri, C. (1980).
On the menbership problem for functional and multivalued dependencies in relational databases.
ACM Trans. Database Syst., 5(3):241–259.

Clarke, E. and Emerson, E. (1981).
Design and synthesis of synchronization skeletons using branching time temporal logic.
In *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71.

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

Emerson, E. and Halpern, J. (1986).
Sometimes and not never revisited: On branching versus linear time temporal logic.
Journal of the ACM, 33(1):151–178.

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

Immerman, N. (1981).
Number of quantifiers is better than number of tape cells.
Journal of Computer and System Sciences, 22(3):384 – 406.

Jones, N. D. (1975).
Space-bounded reducibility among combinatorial problems.
Journal of Computer and System Sciences, 11(1):68 – 85.

Jones, N. D. (1977).
Corrigendum: Space-bounded reducibility among combinatorial problems.
J. Comput. Syst. Sci., 15(2):241.

Lichtenstein, O. and Pnueli, A. (1985).
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. ACM.

Pnueli, A. (1977).
The temporal logic of programs.
In *Proceedings of FOCS*, pages 46–57.

Schnoebelen, P. (2003).
The complexity of temporal model checking.
In *Advances in Modal Logics, Proceedings of AiML 2002*. World Scientific.

Schobbens, P. Y. (2004).
Alternating-time logic with imperfect recall.
Electronic Notes in Theoretical Computer Science, 85(2).

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

Vardi, M. Y. and Wolper, P. (1986).
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.

4. Reasoning about Strategies

4.1 Alternating-Time Temporal Logics

The picture so far...

What kind of logics did we introduce so far?

- *Basic Modal Logics*: modelling what is *possible and necessary*
- Instantiations: *epistemic logics*, dynamic logics, ...
- Linear-time temporal logic (**LTL**)
- Branching-time logics (**CTL** and **CTL***)

In the temporal case each *transition* modelled a *time step*. We considered only "actor".

Now: Modelling abilities of *multiple agents*.

Agents can *execute actions* and *cooperate*. *Action profiles* determine the behaviour of the system.

Alternating-time Temporal Logics

- **ATL, ATL*** [Alur et al. 1997]
- *Temporal logic* meets *game theory*
- Modeling abilities of *multiple agents*
- Main idea: *cooperation modalities*

$\langle\langle A\rangle\rangle\varphi$: *coalition A has a collective strategy to enforce φ*

Enforcement is understood in the game-theoretical sense: There is a *winning strategy*.

The syntax is given as for the computation-tree logics.

Definition 4.1 (Language $\mathcal{L}_{ATL^*}$ [Alur et al., 1997])

The *language $\mathcal{L}_{ATL^*}$* is given by all formulae generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle\gamma \quad \text{where}$$
$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \gamma\,\mathcal{U}\,\gamma \mid \bigcirc\gamma,$$

$A \subseteq \mathbb{A}\mathrm{gt}$, and $p \in \mathcal{P}rop$. Formulae φ (resp. γ) are called *state* (resp. *path*) formulae.

The language $\mathcal{L}_{ATL}$ *restricts* $\mathcal{L}_{ATL^*}$ in the same way as $\mathcal{L}_{CTL}$ restricts $\mathcal{L}_{CTL^*}$: *Each temporal operator must be directly preceded by a cooperation modality*.

Definition 4.2 (Language $\mathcal{L}_{ATL}$ [Alur et al., 1997])

The *language* $\mathcal{L}_{ATL}$ is given by all formulae generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box \varphi \mid \langle\langle A\rangle\rangle \varphi \,\mathcal{U}\, \varphi$$

where $A \subseteq \mathbb{A}\mathrm{gt}$ and $p \in \mathcal{P}rop$.

The language $\mathcal{L}_{ATL^+}$ *restricts* $\mathcal{L}_{ATL^*}$ but extends $\mathcal{L}_{ATL}$. It allows for Boolean combinations of path formulae.

Definition 4.3 (Language $\mathcal{L}_{ATL^+}$)

The *language* $\mathcal{L}_{ATL^+}$ is given by all formulae generated by the following grammar:

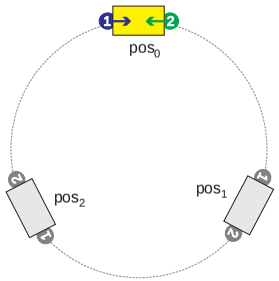
$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \gamma,$$
$$\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \bigcirc \varphi \mid \varphi \,\mathcal{U}\, \varphi.$$

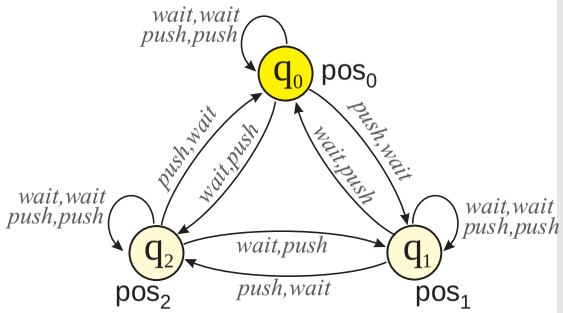
where $A \subseteq \mathbb{A}\mathrm{gt}$ and $p \in \mathcal{P}rop$.

Some example formulae: $\rightsquigarrow$ *blackboard*

ATL Models: Concurrent Game Structures

- **Agents**, **actions**, **transitions**, atomic **propositions**
- Atomic propositions + **interpretation**
- **Actions are abstract**





Definition 4.4 (Concurrent Game Structure)

A *concurrent game structure* is a tuple $\mathfrak{M} = \langle \mathbb{A}\mathrm{gt}, Q, \pi, Act, d, o\rangle$, where:

- $\mathbb{A}\mathrm{gt}$: a finite set of all *agents*;
- Q: a set of *states*;
- $\pi : Q \to \mathcal{P}(\mathcal{P}rop)$: a *valuation* of propositions;
- Act: a finite set of (atomic) *actions*;
- $d : \mathbb{A}\mathrm{gt} \times Q \to \mathcal{P}(Act)$ defines actions *available* to an agent in a state;
- o: a deterministic *transition function* that assigns outcome states $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to states and tuples of actions.

Recall and information

A *strategy* of agent a is a *conditional plan* that specifies what a is going to do in each situation.

Two types of "situations": Decisions are based on

- the *current state* only ($\rightsquigarrow$ *memoryless strategies*)
  $$s_a : Q \to Act.$$
- on the *whole history* of events that have happened ($\rightsquigarrow$ *prefect recall strategies*)
  $$s_a : Q^+ \to Act.$$

We also distinguish between agents with

- *perfect information* (all states are distinguishable).
- *imperfect information* (some state are indistinguishable).

Perfect Information Strategies

Definition 4.5 (*IR*- and *Ir*-strategies)

- A *perfect information perfect recall strategy* for agent a (*IR-strategy* for short) is a function
  $$s_a : Q^+ \to Act \text{ such that } s_a(q_0 q_1 \dots q_n) \in d_a(q_n).$$
  The set of such strategies is denoted by Σ_a^{IR}.

- A *perfect information memoryless strategy* for agent a (*Ir-strategy* for short) is given by a function
  $$s_a : Q \to Act \text{ where } s_a(q) \in d_a(q).$$
  The set of such strategies is denoted by Σ_a^{Ir}.

i (resp. *I*) stands for *imperfect* (resp. *perfect*) *information* and *r* (resp. *R*) for *imperfect* (resp. *perfect*) *recall*. [Schobbens, 2004]

Some Notation

The following holds for all kind of strategies:

- A *collective strategy* for a group of agents $A = \{a_1, \dots, a_r\} \subseteq \mathbb{A}\text{gt}$ is a set
  $$s_A = \{s_a \mid a \in A\}$$
  of strategies, one per agent from A.
- $s_A|_a$, we denote agent a*'s part of the collective strategy* s_A, $s_A|_a = s_A \cap \Sigma_a$.
- $s_\emptyset = \emptyset$ denotes the strategy of the *empty coalition*.
- Σ_A denotes the *set of all collective strategies* of A.
- $\Sigma = \Sigma_{\mathbb{A}\text{gt}}$

Outcome of a strategy

$out(q, s_A)$= set of *all paths* that *may occur* when agents A *execute* s_A *from* state q *onward*.

Definition 4.6 (Outcome)

$\lambda = q_0 q_1 \dots \in Q \in out(q, s_A) \subseteq Q^\omega$ iff

1. $q_0 = q$
2. for each $i = 1, \dots$ there is a tuple $(\alpha_1^{i-1}, \dots, \alpha_k^{i-1}) \in Act^k$ such that
   - $\alpha_a^{i-1} \in d_a(q_{i-1})$ for each $a \in \mathbb{A}\text{gt}$,
   - $\alpha_a^{i-1} = s_A|_a(q_0 q_1 \dots q_{i-1})$ for each $a \in A$, and
   - $o(q_{i-1}, \alpha_1^{i-1}, \dots, \alpha_k^{i-1}) = q_i$.

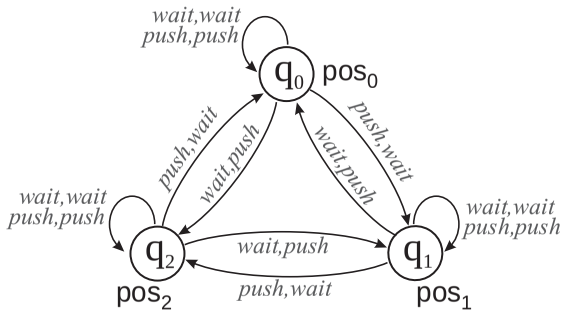
For an *Ir-strategy* replace "$s_A|_a(q_0 q_1 \dots q_{i-1})$" by "$s_A|_a(q_{i-1})$".

Definition 4.7 (Perfect information semantics)

$\mathfrak{M}, q \models_{Ix} p$	iff p is in $\pi(q)$;
$\mathfrak{M}, q \models_{Ix} \varphi \wedge \psi$	iff $\mathfrak{M}, q \models_{Ix} \varphi$ and $\mathfrak{M}, q \models_{Ix} \psi$;
$\mathfrak{M}, q \models_{Ix} \langle\langle A \rangle\rangle \Phi$	iff *there is a collective Ix-strategy* s_A such that, *for each path* $\lambda \in out(q, s_A)$, we have $\mathfrak{M}, \lambda \models_{Ix} \Phi$.
$\mathfrak{M}, \lambda \models_{Ix} \bigcirc \varphi$	iff $\mathfrak{M}, \lambda[1, \infty] \models_{Ix} \varphi$;
$\mathfrak{M}, \lambda \models_{Ix} \Diamond \varphi$	iff $\mathfrak{M}, \lambda[i, \infty] \models_{Ix} \varphi$ for some $i \geq 0$;
$\mathfrak{M}, \lambda \models_{Ix} \Box \varphi$	iff $\mathfrak{M}, \lambda[i, \infty] \models_{Ix} \varphi$ for all $i \geq 0$;
$\mathfrak{M}, \lambda \models_{Ix} \varphi \mathcal{U} \psi$	iff $\mathfrak{M}, \lambda[i, \infty] \models_{Ix} \psi$ for some $i \geq 0$, and $\mathfrak{M}, \lambda[j, \infty] \models_{Ix} \varphi$ forall $0 \leq j \leq i$.

Note that temporal formulae and the Boolean connectives are handled as before.

Example: Robots and Carriage



$pos_0 \rightarrow \langle\langle 1 \rangle\rangle \Box \neg pos_1$

Definition 4.8 (ATL$_{Ix}$, ATL$^{+}_{Ix}$, ATL$^{*}_{Ix}$, ATL, ATL*)

We define **ATL$_{Ix}$**, **ATL$^{+}_{Ix}$**, and **ATL$^{*}_{Ix}$** as the logics $(\mathcal{L}_{ATL}, \models_{Ix})$, $(\mathcal{L}_{ATL^+}, \models_{Ix})$ and $(\mathcal{L}_{ATL^*}, \models_{Ix})$ where $x \in \{r, R\}$, respectively. Moreover, we use **ATL** (resp. **ATL***) as an abbreviation for **ATL$_{IR}$** (resp. **ATL$^{*}_{IR}$**).

Intuitively, a logic is given by the set of all *valid formulae*.

Example 4.9 (A simple scenario)

$\rightsquigarrow$ *blackboard*

Theorem 4.10

For $\mathcal{L}_{ATL}$, the perfect recall semantics is equivalent to the memoryless semantics under perfect information , i.e., $\mathfrak{M}, q \models_{IR} \varphi$ iff $\mathfrak{M}, q \models_{Ir} \varphi$. Both semantics are different for $\mathcal{L}_{ATL^*}$. That is

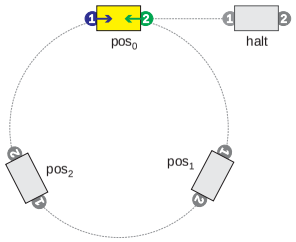
$$ATL = ATL_{Ir} = ATL_{IR}.$$

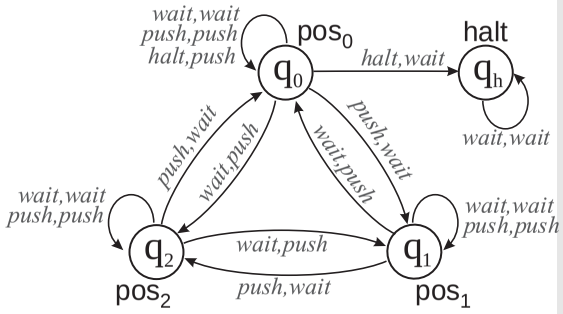
Proof idea.

The first "non-looping part" of each path has to satisfy a formula. $\rightsquigarrow$ Exercise □

The property has been first observed in [Schobbens, 2004] but it follows from [Alur et al., 2002] in a straightforward way.

Example: Robots and Carriage (2)





What about $\langle\langle 1,2\rangle\rangle(\Diamond pos_1 \wedge \Diamond halt)$?

$\mathfrak{M}, q_0 \models_{IR} \langle\langle 1,2\rangle\rangle(\Diamond pos_1 \wedge \Diamond halt)$
$\mathfrak{M}, q_0 \not\models_{Ir} \langle\langle 1,2\rangle\rangle(\Diamond pos_1 \wedge \Diamond halt)$

4.2 Imperfect Information

Imperfect information

How can we reason about agents/extensive games with *imperfect information*?

We combine ***ATL**** and *epistemic logic*.

- We extend CGSs with *indistinguishability relations* $\sim_a \subseteq Q \times Q$, one per agent. The relations are assumed to be *equivalence relations*.
- We interpret $\langle\langle A\rangle\rangle$ epistemically ($\rightsquigarrow \models_{iR}$ and $\models_{ir}$)

$\rightsquigarrow$ *Problems!*

Strategic ability and knowledge are not independent.

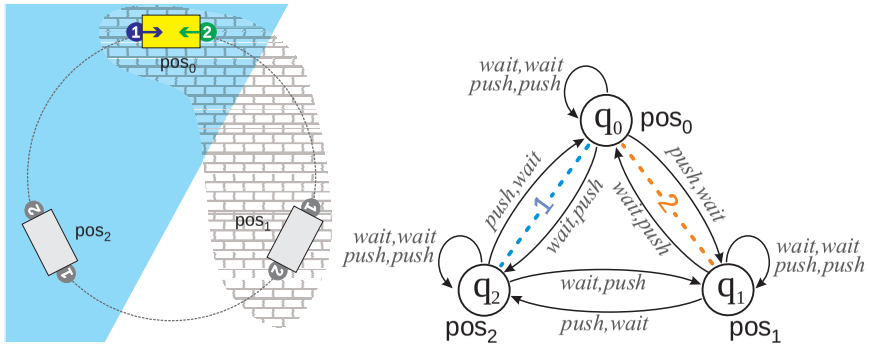
Definition 4.11 (CEGS)

A *concurrent epistemic game structure* (CEGS) is a tuple

$$\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, \{\sim_a \mid a \in \mathbb{A}\mathrm{gt}\}) \quad \text{with}$$

- $(\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o)$ a CGS and
- $\sim_a \subseteq Q \times Q$ equivalence relations (*indistinguishability relations*).

Example: Robots and Carriage



What about $\langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle \bigcirc pos_1$ in q_0?

$\mathfrak{M}, q_0 \models_{Ir} \langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle \bigcirc pos_1$

$\mathfrak{M}, q_0 \not\models_{ir} \langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle \bigcirc pos_1$

Problem:

Strategic and epistemic abilities are *not* independent!

$\langle\langle A \rangle\rangle \Phi$ = A can *enforce* Φ

It should at least mean that A are able to *identify* and *execute* the right strategy!

Executable strategies = *uniform strategies*

Definition 4.12 (Uniform strategy)

Strategy s_a is *uniform* iff it specifies the *same choices for indistinguishable situations* :

- Memoryless strategies:

  $$\text{if } q \sim_a q' \text{ then } s_a(q) = s_a(q').$$

- Perfect recall:

  $$\text{if } \lambda \approx_a \lambda' \text{ then } \Rightarrow s_a(\lambda) = s_a(\lambda'),$$

  where $\lambda \approx_a \lambda'$ iff $\lambda[i] \sim_a \lambda'[i]$ for every i.

A *collective strategy* is uniform iff it consists only of uniform individual strategies.

Imperfect Information Strategies

Definition 4.13 (*IR*- and *Ir*-strategies)

- A *imperfect information perfect recall strategy* for agent a (*iR-strategy* for short) is a *uniform IR-strategy*.
- A *imperfect information memoryless strategy* for agent a (*ir-strategy* for short) is a *uniform Ir-strategy*.

The *outcome* is defined as before.

Imperfect Information Semantics

The *imperfect information semantics* is defined as before, only the clause for

$\mathfrak{M}, q \models_{\text{IX}} \langle\langle A \rangle\rangle \varphi$ iff *there is a collective IX-strategy* s_A such that, for each path $\lambda \in out(q, s_A)$, we have $\mathfrak{M}, \lambda \models_{\text{IX}} \varphi$.

is replaced by

$\mathfrak{M}, q \models_{ix} \langle\langle A \rangle\rangle \varphi$ iff *there is a collective ix-strategy* s_A such that, for each path $\lambda \in \bigcup_{q' : q \sim_A q'} out(q', s_A)$, we have $\mathfrak{M}, \lambda \models_{ix} \varphi$

where $x \in \{r, R\}$ and $\sim_A := \bigcup_{a \in A} \sim_a$.

Remark 4.14

This definition models that "everybody in A knows that φ".

The fixed-point characterisation do not hold anymore!

Theorem 4.15

The following formulae are not valid for ***ATL$_{ir}$****:*

- $\langle\langle A \rangle\rangle \Box \varphi \quad \leftrightarrow \quad \varphi \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box \varphi$
- $\langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2 \quad \leftrightarrow \quad \varphi_2 \vee (\varphi_1 \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2)$.

Proof.

$\rightsquigarrow$: Exercise. □

$\rightsquigarrow$ *blackboard*

4.3 References

Åqvist, L. (1984).
Deontic logic.
In D. M. Gabbay and F. Guenther (Eds.), *Handbook of Philosophical Logic, Vol II*, pp. 605–714. Reidel.

R. Alur, T. A. Henzinger, and O. Kupferman (2002).
Alternating-time Temporal Logic.
Journal of the ACM, 49:672–713.

Broersen, J., Dastani, M., Huang, Z., and van der Torre, L. (2001a).
The BOID architecture: conflicts between beliefs, obligations, intentions and desires.
In Müller, J., Andre, E., Sen, S., and Frasson, C., editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 9–16. ACM Press.

Bulling, N., Dix, J., and Jamroga, W. (2010).
Model checking logics of strategic ability: Complexity.
In Dastani, M., Hindriks, K. V., and Meyer, J.-J. C., editors, *Specification and Verification of Multi-Agent Systems*. Springer.

Cohen, P. and Levesque, H. (1990).
Intention is choice with commitment.
Artificial Intelligence, 42:213–261.

E. A. Emerson (1990).
Temporal and modal logic.
Handbook of Theoretical Computer Science, volume B, 995–1072. Elsevier.

Fagin, R., Halpern, J. Y., Moses, Y. & Vardi, M. Y. (1995).
Reasoning about Knowledge.
MIT Press: Cambridge, MA.

Fausto Giunchiglia and Paolo Traverso (2000).
In Susanne Biundo and Maria Fox (Eds.) *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, 1–20, LNCS 1809, Springer 2000.

Hindriks, K. V., F. S. de Boer, H. Hoek, and J.-J. C. Meyer (1998).
Formal semantics of the core of AGENT-0.
In *ECAI'98 Workshop on Practical Reasoning and Rationality*, pp. 20–29.

Huth, M. & Ryan, M. (2000).
Logic in Computer Science: Modeling and reasoning about systems.
Cambridge University Press.

Jamroga, W. (2004).
Strategic planning through model checking of ATL formulae.
In L. R. et al., editors, *Artificial Intelligence and Soft Computing: Proceedings of ICAISC'04*, volume 3070 of *Lecture Notes in Computer Science*, pages 879–884. Springer Verlag.

Wojtek Jamroga and Jürgen Dix (2005).
Model Checking Strategic Abilities of Agents under Incomplete Information.
In Mario Coppo, Elena Lodi and G. Michele Pinna (Eds.), Proceedings of the Italian Conference on Theoretical Computer Science (ICTCS '05), pages 295–308. LNCS 3701. Springer, 2005.

Wojtek Jamroga and Jürgen Dix (2005).
Do Agents Make Model Checking Explode (Computationally)?
M. Pechoucek and P. Petta and L.Z. Varga (Eds.), Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS '05), pages 398-407. LNCS 3690. Springer, 2005.

Wojtek Jamroga and Jürgen Dix. (2008)
Model Checking Abilities of Agents: A closer look.
Journal of Computing Systems, **42(3)**, 366–410.

Kripke, S. (1963a).
Semantical analysis of modal logic I. Normal propositional calculi.
Zeitschrift fur math. Logik und Grundlagen der Mathematik 9, 67–96.

Kripke, S. (1963b).
Semantical considerations on modal logic.
Acta Philosophica Fennica 16, 83–94.

Kripke, S. (1965).
Semantical analysis of modal logic II. Non-normal modal propositional calculi.
In Addison, Henkin, and Tarski (Eds.), *The theory of models*, Amsterdam, North-Holland, pp. 206–220.

Pistore, M. and Traverso, P. (2001).
Planning as model checking for extended goals in non-deterministic domains.
In *Proceedings of IJCAI*, pages 479–486.

Rao, A. S. and M. Georgeff (1991).
Modeling Rational Agents within a BDI-Architecture.
In J. F. Allen, R. Fikes, and E. Sandewall (Eds.), *Proceedings of the International Conference on Knowledge Representation and Reasoning*, Cambridge, MA, pp. 473–484. Morgan Kaufmann.

von Wright, G. H. (1951).
Deontic logic.
Mind 60, 1–15.
Reprinted in G. H. von Wright, *Logical Studies*, pp. 58–74. Routledge and Kegan Paul, 1957.

Wooldridge, M. (2000).
Reasoning about Rational Agents.
MIT Press : Cambridge, Mass.

5. Model Checking Strategic Logics

Overview

- In this section we turn to model checking *strategic logics* (more precisely, *alternating-time temporal logics*).
- The results make *use of techniques introduced for **LTL** and **CTL****.
- We assume familiarity with *basic concepts of complexity theory*.

5.1 Complexity Theory

Complexity Classes

Deterministic Turing machine (DTM)

- *infinite* (readable and writable) *tape*
- *finitely* many *states*
- *deterministic moves*

Non-deterministic Turing machine (NTM)

Like a DTM but *non-deterministic moves* are allowed.

Orcale Machine (OTM)

- Let A be a *language* . An *A-oracle machine* is a DTM or NTM with a *subroutine* which allows to decide in *one step* whether $w \in A$ for some word w.
- For a *complexity class $\mathcal{C}$* a *$\mathcal{C}$-oracle* machine is a *A-oracle machine* for *any* $A \in \mathcal{C}$.

Complexity Classes $\Sigma_2^{\mathbf{P}}$, $\Delta_2^{\mathbf{P}}$, $\Delta_3^{\mathbf{P}}$

- $\Sigma_{\mathbf{i}}^{\mathbf{P}}$: problems solvable in *polynomial time* by a *non-deterministic* Turing machine making adaptive queries to a $\Sigma_{\mathbf{i-1}}^{\mathbf{P}}$ oracle; i.e. by $\Sigma_{\mathbf{i-1}}^{\mathbf{P}}$*-oracle polynomial time NTMs*.
- $\Sigma_{\mathbf{2}}^{\mathbf{P}} = \mathrm{NP}^{\mathrm{NP}}$: problems solvable in *polynomial time* by a *non-deterministic* Turing machine making adaptive queries to an *NP* oracle.
- $\Delta_{\mathbf{2}}^{\mathbf{P}} = \mathrm{P}^{\mathrm{NP}}$: A problem is in $\Delta_2^P = \mathbf{P}^{\mathbf{NP}}$ if it can be solved in *deterministic polynomial time* with subcalls to an $\mathbf{NP}$-oracle. We also have $\Delta_3^P := \mathbf{P}^{[\mathbf{NP}^{\mathbf{NP}}]}$ and $\Delta_1^P = \mathbf{P}$.

We have:

$$\mathbf{P} = \Delta_1^P \subseteq \Sigma_1^P = \mathbf{NP} \subseteq \Delta_2^P \subseteq \Sigma_2^P \subseteq \cdots \subseteq PH \subseteq PSPACE.$$

5.2 Types of Stategies

Strategies

We have introduced *four types of strategies*:

1. *ir*-strategies;
2. *Ir*-strategies;
3. *IR*-strategies;
4. *iR*-strategies.

How many strategies are there for each type?

1. exponentially many;
2. exponentially many;
3. infinitely many;
4. infinitely many.

Exponentially many wrt the size of the input! $\approx |\mathbf{Act}|^{|\mathbf{Q}|}$

Assume we are looking for a *"good" Ir-strategy* wrt some *property* P. How complex is this task? (Upper bound)

It is in $\mathbf{NP}$, provided $P \in \mathbf{P}$!

1. *Guess* s_A;
2. *check* whether s_A *satisfies* P.

And the case for *"good" ir-strategies*?

It is also in $\mathbf{NP}$, provided $P \in \mathbf{P}$! Why? What about uniformity?

1. *Guess Ir*-strategy s_A;
2. *check* whether it is an *ir*-strategy, i.e. for uniformity (Q is finite!);
3. *check* whether s_A *satisfies* P.

What if P is *verifiable in $\mathcal{C}$* for an *arbitrary complexity class $\mathcal{C}$*?

Finding *ir*- and *Ir*-strategies is in $\mathbf{NP}^{\mathcal{C}}$.

What about *perfect recall* strategies?

There are *infinitely* many: So there is *no general method*!

5.3 ATL

Perfect Information

Recall Theorem 4.10: ***ATL* = *ATL*$_{Ir}$ = *ATL*$_{IR}$**

The **ATL** model checking algorithm employs the well-known *fixpoint characterisations* :

$$\langle\langle A\rangle\rangle\Box\varphi \leftrightarrow \varphi \wedge \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\varphi,$$
$$\langle\langle A\rangle\rangle\varphi_1\,\mathcal{U}\,\varphi_2 \leftrightarrow \varphi_2 \vee \varphi_1 \wedge \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\varphi_1\,\mathcal{U}\,\varphi_2.$$

Do these characterisations also hold for incomplete information?

No! A choice of an action at a state q has *non-local consequences*: It automatically *fixes choices* at *all states q' indistinguishable from q* for the coalition A.

function $mcheck(M,\varphi)$.
Returns states q with $\mathfrak{M}, q \models \varphi$.
case $\varphi \in \Pi$: return $\pi(p)$
case $\varphi = \neg\psi$: return $Q \setminus mcheck(M,\psi)$
case $\varphi = \psi_1 \vee \psi_2$: return $mcheck(M,\psi_1) \cup mcheck(M,\psi_2)$
case $\varphi = \langle\langle A\rangle\rangle\bigcirc\psi$: return $pre(M,A,mcheck(M,\psi))$
case $\varphi = \langle\langle A\rangle\rangle\Box\psi$:
  $Q_1 := Q$; $Q_2 := mcheck(M,\psi)$; $Q_3 := Q_2$;
  while $Q_1 \not\subseteq Q_2$
  do $Q_1 := Q_2$; $Q_2 := pre(M,A,Q_1) \cap Q_3$ **od**;
  return Q_1
case $\varphi = \langle\langle A\rangle\rangle\psi_1\,\mathcal{U}\,\psi_2$:
  $Q_1 := \varnothing$; $Q_2 := mcheck(M,\psi_1)$;
  $Q_3 := mcheck(M,\psi_2)$;
  while $Q_3 \not\subseteq Q_1$
  do $Q_1 := Q_1 \cup Q_3$; $Q_3 := pre(M,A,Q_1) \cap Q_2$ **od**;
  return Q_1
end case

Multi-agent extension of **CTL** model checking.

function $pre(M, A, Q)$.
Auxiliary function; returns the exact set of *states Q' such that*, when the system is in a state $q \in Q'$, agents A *can cooperate and enforce the next state to be in* Q.
return $\{q \mid \exists \alpha_A \forall \alpha_{\mathbb{A}gt \setminus A}\ o(q, \alpha_A, \alpha_{\mathbb{A}gt \setminus A}) \in Q\}$

The function follows the same idea as the pre-image function of **CTL** model checking.

Theorem 5.1 (ATL$_{Ir}$ and ATL$_{IR}$ [Alur et al., 2002])

*Model checking **ATL$_{Ir}$** and **ATL$_{IR}$** is $\mathbf{P}$-complete, and can be done in time $\mathbf{O}(|\mathfrak{M}| \cdot |\varphi|)$, where $|\mathfrak{M}|$ is given by the number of transitions in $\mathfrak{M}$.*

Note, that the size of $\mathfrak{M}$ is *exponential* in the number of *states* and *agents*!

Proof: Upper Bound

- Each case of the algorithm is called at most $\mathbf{O}(|\varphi|)$ times and terminates after $\mathbf{O}(|\mathfrak{M}|)$.
- The latter is shown by translating the model to a two-player game [Alur et al., 2002], and then solving the "invariance game" on it in polynomial time ([Beeri, 1980], [Alur et al., 2002]).

And-Or-Graph Reachability

For the *lower bound*, we *reduce reachability in and-or-graphs*.

An *and-or graph* [Immerman, 1981]

- is a tuple (E, V, l) such that $G = (E, V)$ is a *directed acyclic* graph and $l : V \to \{\wedge, \vee\}$ *a labeling function*.

Let $x_1, \ldots, x_n$ denote all *successor nodes of* u. v is said to be *reachable* from u iff

1. $u = v$; or
2. $l(u) = \wedge$, $n \geq 1$, and v is *reachable* from all x_i's; or,
3. $l(u) = \vee$, $n \geq 1$, and v is *reachable* from some x_i.

Theorem 5.2 ([Immerman, 1981])

The and-or-graph reachability problem is $\mathbf{P}$-complete.

Proof: Lower Bound

Hardness is shown by a reduction of reachability in And-Or-Graphs:

- Transform and-or-graph to a CGS;
- Player 1 owns or-states;
- Player 2 owns and-states;
- v *reachable from* a *iff* $\mathfrak{M}, a \models \langle\langle 1 \rangle\rangle \Diamond l_v$.

Imperfect Information

Agent's ability to *identify* a strategy as winning also varies throughout the game in an arbitrary way (agents can learn as well as forget). This suggests that winning strategies *cannot be synthesized incrementally*.

How to model check a formula $\mathfrak{M}, q \models \langle\langle A \rangle\rangle \gamma$ where γ includes *no nested cooperation modalities* ?

Theorem 5.3 (ATL_{ir})

Model checking ***ATL_{ir}*** *is Δ_2^P-complete.*

Proof: Lower Bound

Reduction of SNSAT_1.

Recall: $\Delta_2^P = P^{\text{NP}}$

Proof: Upper Bound

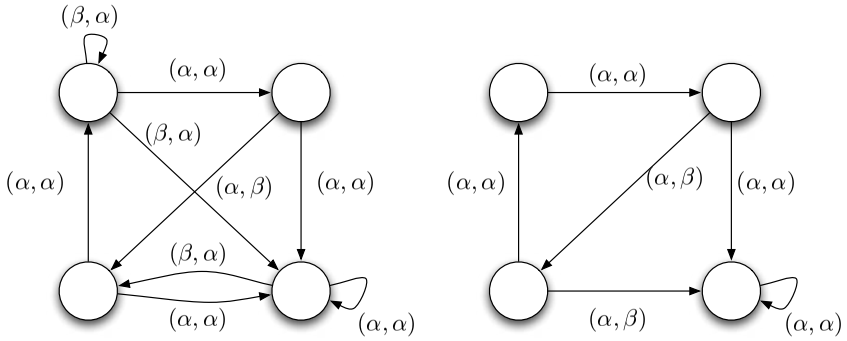
Let $\langle\langle A \rangle\rangle \gamma$ be given where γ includes no nested cooperation modalities.

1. *Guess a strategy* s_A of A.
2. " *Prune*" $\mathfrak{M}$ to $\mathfrak{M}|_{s_A}$; i.e. remove transitions that cannot occur according to s_A.
3. *Remove labels* from $\mathfrak{M}|_{s_A}$ and interpret it as *Kripke structure* $\mathfrak{M}'|_{s_A}$
4. Then,

$$\mathfrak{M}, q \models \langle\langle A \rangle\rangle \gamma \text{ iff } \mathfrak{M}'|_{s_A}, q \models^{\text{CTL}} \text{A}\gamma$$

The basic idea is to *guess* a strategy and apply ***CTL*** *model checking*.

ATL and CTL: Pruning



Guess the strategy s_1 in which 1 always *plays* α .

$\langle\langle 1 \rangle\rangle \Diamond \gamma \quad \rightsquigarrow \quad$ *guess* s_1 , check $\text{A}\Diamond\gamma$ in the *pruned* model

5.4 ATL*

Tree automata

- *ω-automata* accept *infinite words*.
- **LTL** formulae are interpreted over infinite words.
- Fixing a *strategy* and *unraveling* a CGS results in an infinite *tree*.
- *Tree automata on infinite trees* accept *infinite trees* instead of *infinite words*.
- The *transition relation* of a tree automata accepting trees with maximal *branching* k is given by

$$\Delta : Q \times \Sigma \times \{1, \ldots, k\} \to \mathcal{P}(\cup_{i=1\ldots k} Q^i)$$

with $\Delta(q, a, i) \subseteq Q^i$.

- A tree is *accepted* if each branch of the tree is accepted in the same way as for ω-automata.

Theorem 5.4 (ATL^*_{ir} and ATL^*_{Ir} [Schobbens, 2004])

*Model checking **ATL^*_{ir}** and **ATL^*_{Ir}** is $PSPACE$-complete in the number of transitions in the model and the length of the formula.*

Proof: Lower Bound

$\mathcal{L}_{LTL}$ is contained in $\mathcal{L}_{ATL^*}$ which renders $\mathcal{L}_{ATL^*}$ with the perfect information memoryless semantics to be *at least $PSPACE$-hard*.

Proof: Upper Bound

Let $\langle\langle A \rangle\rangle \psi$ where ψ is an $\mathcal{L}_{LTL}$-formula.

1. *Guess* an *ir*-strategy s_A of A.
2. "*Prune*" $\mathfrak{M}$ to $\mathfrak{M}|_{s_A}$; i.e. remove transitions that cannot occur according to s_A.
3. *Remove labels* from $\mathfrak{M}|_{s_A}$ and interpret it as Kripke structure $\mathfrak{M}'|_{s_A}$
4. Then,

$$\mathfrak{M}, q \models \langle\langle A \rangle\rangle \gamma \text{ iff } \mathfrak{M}'|_{s_A}, q \models^{CTL^*} \mathsf{A}\gamma$$

This procedure can be performed in $\mathbf{NP}^{PSPACE}$, which renders the complexity of the whole language to be in $\mathbf{P}^{\mathbf{NP}^{PSPACE}} = PSPACE$.

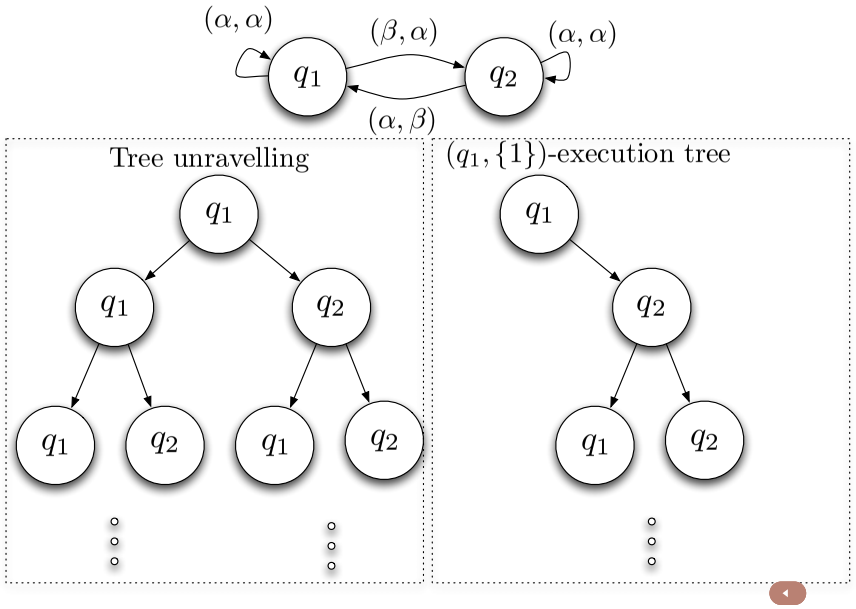
Theorem 5.5 (ATL^*_{IR} [Alur et al., 2002])

*Model checking **ATL^*_{IR}** is $2EXPTIME$-complete in the number of transitions in the model and the length of the formula.*

Proof sketch

Let $\mathfrak{M}$ be a CGS and $\langle\langle A \rangle\rangle \psi$ be an $\mathcal{L}_{ATL^*}$-formula (where we assume that ψ is an $\mathcal{L}_{LTL}$-formula).

- Given s_A of A and a state q. *Unfold model* into a *q-rooted tree* representing all *possible behaviors with agents A following their strategy s_A*.
- *(q, A)-execution tree* is *induced* by $out(q, s_A)$.



Proof sketch: Upper Bound

1. Construct Büchi *tree automaton* $A_{\mathfrak{M},q,A}$ that *accepts exactly the* (q,A)*-execution trees*
2. Construct a *Rabin tree automaton* which accepts all trees that satisfy the $\mathcal{L}_{CTL^*}$-formula $A\psi$ [Emerson and Sistla, 1984].
3. product automaton $A_\psi \times A_{\mathfrak{M},q,A}$, accepting the trees accepted by both automata, is a *Rabin tree automaton* with $n := \mathbf{O}(|A_\psi| \cdot |A_{\mathfrak{M},q,A}|)$ many states and $r := 2^{\mathbf{O}(|\psi|)}$ many Rabin pairs
4. Emptiness check can be done in time $\mathbf{O}(n \cdot r)^{3r}$

Lower Bound: *realizability of* ***LTL****-formulae*
[Pnueli and Rosner, 1989, Rosner, 1992, Alur et al., 2002].

5.5 ATL$^+$

Theorem 5.6 (ATL^+_{ir} and ATL^+_{Ir} [Schobbens, 2004])

Model checking ***ATL****$^+_{ir}$ and* ***ATL****$^+_{Ir}$ is* Δ_3*-complete in the number of transitions in the model and the length of the formula.*

Proof

1. *Guess* a strategy s_A of A.
2. "*Prune*" $\mathfrak{M}$ to $\mathfrak{M}|_{s_A}$.
3. *Remove labels* from $\mathfrak{M}|_{s_A} \rightsquigarrow \mathfrak{M}'|_{s_A}$.
4. Then, $\mathfrak{M}, q \models \langle\langle A \rangle\rangle \gamma$ *iff* $\mathfrak{M}'|_{s_A}, q \models^{CTL^+} A\gamma$.

Complexity: $\Delta_2^{P\Delta_2^P} = \Delta_3^P$.
Hardness: Reduction of the SNSAT$_3$.

The following result of the upper bound is more sophisticated. The idea is that only a *finite segment of fixed length of an Ir-strategy is needed*. For the lower bound the *QSAT problem* is reduced to model checking $\mathbf{ATL}^+_{IR}$.

Theorem 5.7 ($\mathbf{ATL}^+_{IR}$ [Bulling and Jamroga, 2010])

Model checking $\mathbf{ATL}^+_{IR}$ is $PSPACE$-complete wrt the

- *size of the model and*
- *length of the formula*

(Even for turn-based models with two agents and "flat" $\mathcal{L}_{ATL^+}$ formulae.)

Proof of The Lower Bound

We prove the *$PSPACE$-hardness by a reduction of Quantified Boolean Satisfiability (QSAT)*, a canonical $PSPACE$-complete problem.

Definition 5.8 (QSAT)

Input: A *Boolean formula Φ* (in negation normal form)[a] with n *propositional variables* $x_1, \ldots, x_n$.

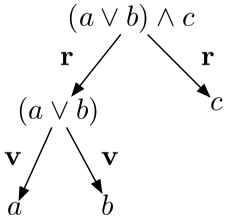
$$\textbf{Output is } \begin{cases} \text{true}, & \text{if } \exists x_1 \forall x_2 \ldots Q_n x_n \ \Phi \text{ is satisfiable,} \\ \text{false}, & \text{otherwise.} \end{cases}$$

(Where $Q_n = \forall$ if n is even, and $Q_n = \exists$ if n is odd.)

[a] That is, negations occur only at literals.

Proof of The Lower Bound

- Model as a game between *verifier* $\mathbf{v}$ and *refuter* $\mathbf{r}$.
- Verifier chooses *disjunctions*, refuter *conjunctions*.

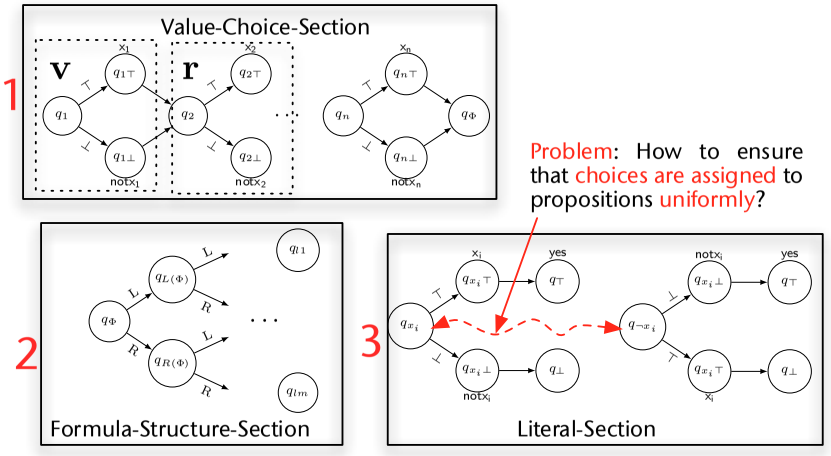


The reduction proceeds in three stages:

1. *Value-coice-section*: $\mathbf{v}$ and $\mathbf{r}$ assign a value to "their" variables

$$\underbrace{\exists x_1}_{\mathbf{v}} \underbrace{\forall x_2}_{\mathbf{r}} \ldots \underbrace{\exists/\forall x_n}_{\mathbf{v/r}}$$

2. *Formula-structure-section*: The result of the "game" determines a literal.
3. *Literal-section*: Check wether the literal is consistent with the value of the variable.



$Cons_i \equiv \Box\neg x_i \vee \Box\neg notx_i$: x_i cannot be declared both $\top$ and $\bot$ during a single execution

Lemma 5.9

$\exists x_1 \forall x_2 \ldots Q_n x_n \; \Phi$ *iff*

$$\mathfrak{M}, q_1 \models_{IR} \langle\langle \mathbf{v} \rangle\rangle \Big(\bigwedge_{i \in Odd} Cons_i \wedge \Big(\bigwedge_{i \in Even} Cons_i \rightarrow \Diamond yes \Big) \Big).$$

Where is the perfect recall needed?

5.6 Imperfect Information and Perfect Recall

Conjecture 1 (ATL_{iR})

Model checking ***ATL_{iR}*** *is undecidable*.

Recently, a proof has been proposed by Dima and Tiplea (June 2010).

Conjecture 2 (ATL^*_{iR})

Model checking ***ATL^*_{iR}*** *is undecidable*.

Conjecture 3 (ATL^+_{iR})

Model checking ***ATL^+_{iR}*** *is undecidable*.

5.7 Summary

So, let's model-check!

Not as easy as it seems.

Complexity of Model Checking CTL and ATL

- Nice results: *model checking CTL and ATL is tractable.*
- But: the result is *relative to the size of the model and the formula*
- Well known catch (CTL): *size of models is exponential* wrt a higher-level description
- Another problem: transitions are labelled
- So: *the number of transitions can be exponential in the number of agents.*

Summary of Complexity Results

	Ir	*IR*	*ir*	*iR*
$\mathcal{L}_{ATL}$	$\mathbf{P}$	$\mathbf{P}$	Δ_2^P	Undecidable†
$\mathcal{L}_{ATL^+}$	Δ_3^P	$PSPACE$	Δ_3^P	Undecidable†
$\mathcal{L}_{ATL^*}$	$PSPACE$	$2EXPTIME$	$PSPACE$	Undecidable†

Figure 4: † These problems are believed to be undecidable.

5.8 References

Alur, R., Henzinger, T. A., and Kupferman, O. (2002).
Alternating-time Temporal Logic.
Journal of the ACM, 49:672–713.

Bulling, N., Dix, J., and Jamroga, W. (2010).
Model checking logics of strategic ability: Complexity.
In Dastani, M., Hindriks, K. V., and Meyer, J.-J. C., editors, *Specification and Verification of Multi-Agent Systems*. Springer.

Bulling, N. and Jamroga, W. (2010).
Verifying agents with memory is harder than it seemed.
In *Proceedings of AAMAS 2010*, pages 699–706, Toronto, Canada. ACM Press.

Schobbens, P. Y. (2004).
Alternating-time logic with imperfect recall.
Electronic Notes in Theoretical Computer Science, 85(2).

Emerson, E. A. and Sistla, A. P. (1984).
Deciding branching time logic.
In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 14–24, New York, NY, USA. ACM.

Pnueli, A. and Rosner, R. (1989).
On the synthesis of a reactive module.
In *POPL '89: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 179–190, New York, NY, USA. ACM.

Rosner, R. (1992).
Modular Synthesis of Reactive Systems.
PhD thesis, Weizmann Institute of Science.