Temporal Logics for Specification and Verification

Valentin Goranko
DTU Informatics

FIRST Autumn School on Modal Logic
November 11, 2009

Transition systems

(Labelled) transition system (TS): $\mathcal{T} = \langle S, \{R_a\}_{a \in A} \rangle$, where:

- S is a non-empty set of states (program states, control states, configuration states, memory registers, etc.)
- $\{R_a\}_{a \in A}$ is a non-empty set of binary transition relations on S, labeled by a set A of actions (processes).
  A transition acts, possibly non-deterministically, on each state from S and produces its successor states.

A transition system may also include various types of designated states, such as initial, accepting, terminating, deadlock, etc. states.

All transitions will be assumed total (serial), i.e., every state has at least one successor.

Usually, we will consider simple transition systems $\langle S, R \rangle$ with only one transition relation.

Transition systems everywhere

Real (physical) examples: clocks, vending machines, payphones, semaphores, lifts, etc.

Abstract examples: finite state machines (in particular, finite automata), as well as (configuration graphs of) various other abstract computing devices, such as pushdown automata, Turing machines, Petri nets, counter systems, timed automata, etc.

Abstractly, transition systems are just Kripke frames.

Interpreted transition systems

Interpreted transition system (ITS):

$$\mathcal{M} = (\langle S, \{R_a\}_{a \in A} \rangle, L)$$

$\langle S, \{R_a\}_{a \in A} \rangle$ is a TS and the state description $L : S \to \mathbf{2}^{\mathrm{PROP}}$ is a mapping assigning to each state s a subset $L(s)$ of a fixed set PROP of atomic propositions that are true at s – the label of s.

Using special atomic propositions, labels can be used to identify initial, accepting, etc. special states.

Thus, interpreted transition systems are simply Kripke models.

Instead of labeling, *valuations* $V : \mathrm{PROP} \to \mathbf{2}^S$ are often used in modal logic.

Paths and computations in transition systems

We will only consider transition systems with serial transition relations, i.e. without terminal (deadlock) states. If necessary, that can be achieved by converting all terminal states to *idle states*.

Path (run, execution) in a transition system $\langle S, R \rangle$ is a sequence

$$s_0, s_1, s_2 \ldots,$$

such that $s_i R s_{i+1}$ for every $i \in \mathbb{N}$.

Computation (trace) in an interpreted transition system $\langle S, R, L \rangle$ is the *observable effect of a run:*

$$L(s_0), L(s_1), L(s_2), \ldots$$

Properties of computations: local properties

Local properties: those that refer to immediate successors or predecessors of the current state. Examples:

Some/every immediate successor state satisfies the property φ:

- The system may enable the process τ at the next state.
- If the light was red at the previous state and is orange now, it must turn green at the next state.

Some/every immediate predecessor satisfies the property φ.

Typically, these are conditional, e.g.:

- If the process τ is currently enabled, the scheduller must have disabled the process ρ at the previous state.
- If train will be entering the tunnel at the next moment, the semaphore on the other end of the tunnel must be red now.

Local properties can be iterated a fixed number of times, but not indefinitely.

Universal properties of computations: invariance, safety

Invariance properties are properties that *must always hold throughout the computation*, while safety properties describe events that *must never happen throughout the computation*.
Examples:

- Safety:
- No deadlock will ever occur. I.e.:
- At least one process will be enabled at any moment of time.
- Not more than one process will ever be in its critical section (e.g., not more than one train will ever be in the tunnel) at the same time.
- The temperature of the reactor must never exceed 1000°C.

Also, partial correctness properties:
If a pre-condition P holds at all initial states, then a post-condition Q will/must hold at all accepting (terminating) states.

Existential properties of computations: eventualities, liveness

Eventuality, liveness properties: those that *will (must) happen sometime during the computation*. Examples:

- The execution of the program will terminate.
- If the train has entered the tunnel, it will eventually leave it.
- Once a printing job is activated, eventually it will be completed.
- If a message is sent, eventually it will be delivered.

Also, total correctness properties:
If a pre-condition P holds at the initial state,
then the computation will reach an accepting (terminating) state, where the post-condition Q will hold.

Properties of computations: fairness, precedence

Fairness properties: *All processes will be treated 'fairly' by the operating system (scheduler, etc.)*

Examples:

- Weak fairness: Every continuous request is eventually granted.
- Strong fairness: If a request is repeated infinitely often then it is eventually granted.
- Impartiality: Every process is scheduled infinitely often.

Precedence:

- The event φ will occur before the event ψ, which may or may not occur at all.
- If the train has entered the tunnel, it will eventually leave it, before any other train has entered the tunnel.

Computational tasks in transition systems

- Local model checking: given an ITS $\mathcal{M}$, state s, and a state property P, determine if s satisfies P.
  Likewise for path properties.
- Model satisfiability checking: given an ITS $\mathcal{M}$ and a state property P, determine if there is a state s in $\mathcal{M}$ satisfying P.
- Global model checking: given a state property P, determine the set of states that satisfy P.
- More general: Query evaluation: given a state query, determine the state relation that satisfy it.
  Example of a binary query:
  'The set of all pairs of states (u, w) such that w satisfies the property P and is reachable from u'.
- Satisfiability testing: given a state property P, determine if there is an ITS $\mathcal{M}$ and a state s in $\mathcal{M}$ satisfying P.

Basic modal logic for transition systems

With every type $\tau = (A, \mathrm{PROP})$ of interpreted transition systems we associate a multimodal language $\mathrm{TL} = \mathrm{TL}_\tau$ with a set of atomic propositions PROP and a family of modal operators $(\mathsf{EX}_a)_{a \in A}$, each representing a transition $a \in A$.

Inductive definition of formulae:

$$\varphi = \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{EX}_a\varphi$$

The other logical connectives: $\bot$, $\rightarrow$, $\vee$, $\leftrightarrow$ are defined as usual. The dual of each modal operator EX_a is defined as $\mathsf{AX}_a := \neg\,\mathsf{EX}_a\neg$.

Thus, EX_a corresponds to $\Diamond$ and AX_a corresponds to $\Box$.

This is the basic multi-modal logic denoted here by TL.

TL is not very expressive. In particular, it can only express local properties but no reachability properties; in particular, no safety, eventualities, or fairness.

Kripke semantics of TL

The semantics of TL_τ is the standard Kripke semantics in interpreted transition systems.

The basic semantic notion of truth of a formula at a state s of an interpreted transition system $\mathcal{M} = (S, \{\overset{a}{\longmapsto}\}_{a \in A}, L)$ is defined inductively as follows.

- $\mathcal{M}, s \models \top$;
- $\mathcal{M}, s \models p$ iff $p \in L(s)$;
- $\mathcal{M}, s \models \neg\varphi$ iff $\mathcal{M}, s \not\models \varphi$;
- $\mathcal{M}, \pi \models \varphi \wedge \psi$ if $\mathcal{M}, \pi \models \varphi$ and $\mathcal{M}, \pi \models \psi$;
- $\mathcal{M}, s \models \mathsf{EX}_a\varphi$ if $\mathcal{M}, t \models \varphi$ for some $t \in S$ such that $s \overset{a}{\longmapsto} t$.

The derived truth definition of AX_a becomes:

- $\mathcal{M}, s \models \mathsf{AX}_a\varphi$ if $\mathcal{M}, t \models \varphi$ for every $t \in S$ such that $s \overset{a}{\longmapsto} t$.

Truth, validity, satisfiability

Given an ITS $\mathcal{M}$, state r in $\mathcal{M}$, and a formula φ of TL, we say that φ is:

- satisfied at r in $\mathcal{M}$ if $\mathcal{M}, r \models \varphi$.
  We then also say that $(\mathcal{M}, r)$ is a model of φ.
- satisfiable, if it is satisfied in some ITS.
- valid in $\mathcal{M}$, denoted $\mathcal{M} \models \varphi$, if $\mathcal{M}, s \models \varphi$ for every state $s \in \mathcal{M}$.
- valid, denoted $\models \varphi$, if it is valid in every ITS.

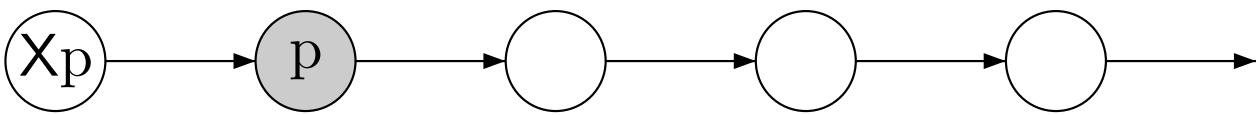
The linear time temporal logic LTL: introduction

- First introduced and studied by Gabbay, Pnueli, Shelah and Stavi in 1980.
- Extends propositional logic with temporal operators for discrete linear time.
- Used to reason about properties of single computations.

Temporal operators in LTL: Nexttime

Whereas φ states a property of the current state, $\mathsf{X}\varphi$ states that the next state satisfies φ.

$\mathsf{X}\mathrm{p}$: nexttime p

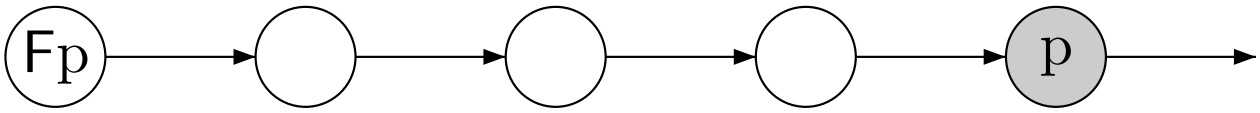


Thus, $\varphi \vee \mathsf{X}\varphi$ states that φ is satisfied now or at the next state.

Temporal operators in LTL: Sometime

Fp claims that some future (or possibly, the current) state satisfies φ without specifying which one, i.e., that "φ will be true sometime in the future".

Fp: sometime p

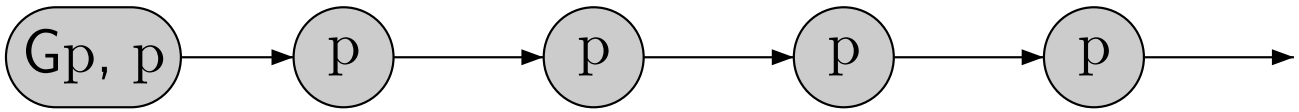


For instance, `alert` $\rightarrow$ F `halt` means that if the system currently is in a state of alert, then it will sometime later be in a halt state.

Temporal operators in LTL: Always

Gφ claims that all the future states (including the current one) satisfy φ, i.e., that "φ will always be true".

Gp: always p



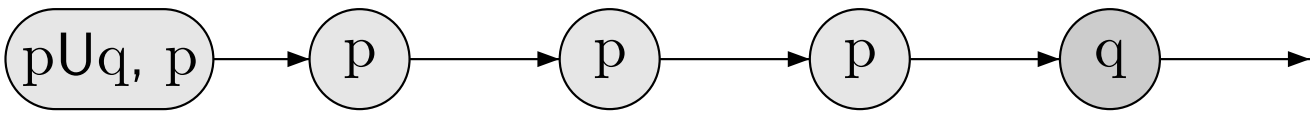
Thus, G(`alert` $\Rightarrow$ F `halt`) means that whenever in the future the system is in a state of alert, it will sometime later be in a halt state.

G is the dual of F: Gφ and $\neg$F$\neg\varphi$ are equivalent.

Temporal operators in LTL: Until

The binary operator U is more expressive than the operator F. The formula $\varphi_1 \mathrm{U} \varphi_2$ states that φ_1 is true until φ_2 is true. More precisely: φ_2 will be true at some future state, and φ_1 will hold in the meantime.

pUq: p until q



The example $\mathrm{G}(\texttt{alert} \rightarrow \mathrm{F}\ \texttt{halt})$ can be refined with the statement that "starting from a state of alert, the alarm remains activated until the halt state is eventually reached":

$$\mathrm{G}(\texttt{alert} \rightarrow (\texttt{alarm}\ \mathrm{U}\ \texttt{halt})).$$

Note that the operator F is a special case of U: $\mathrm{F}\varphi$ and $\texttt{true}\mathrm{U}\varphi$ are equivalent.

Temporal operators in LTL: Other operators

Weak until: W. Intuitively, the statement $\varphi_1 \mathsf{W} \varphi_2$ still expresses "φ_1 until φ_2", but without the inevitable occurrence of φ_2; if φ_2 never occurs, then φ_1 must remain true forever.

Thus, $\varphi_1 \mathsf{W} \varphi_2$ is equivalent to $\mathsf{G}\varphi_1 \vee (\varphi_1 \mathsf{U} \varphi_2)$.

Release: R. Defined as the dual of U, i.e., $\varphi_1 \mathsf{R} \varphi_2 := \neg(\neg\varphi_1 \mathsf{U} \neg\varphi_2)$.

$\varphi_1 \mathsf{R} \varphi_2$ intuitively states that the truth of φ_1 releases the requirement for the truth of φ_2.
Exercise: determine the precise meaning of R.

More precisely, it means that, unless φ_2 is true in all future states, φ_1 must be true sometime in the future and φ_2 must hold between the current state and that future state.

Formulae and syntax of LTL

LTL formulae:

$$\varphi ::= \overbrace{\bot \mid \mathrm{p} \mid \neg\varphi \mid \varphi \wedge \psi}^{\text{propositional logic}} \mid \overbrace{\mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi \mid \varphi \mathsf{U} \psi}^{\text{temporal extension}}$$

where p ranges over a countable set of atomic propositions $\mathrm{AP} = \{p_0, p_1, ...\}$, obtained by abstracting properties.

Since a formula only contains finitely many atomic propositions, whenever suitable we can assume that AP is finite.

F and G can be regarded as definable in terms of U.

Some abbreviations:

$\varphi_1 \rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$,

$\mathsf{F}^\infty\varphi := \mathsf{GF}\varphi$ ("φ holds infinitely often").

$\mathsf{G}^\infty\varphi := \mathsf{FG}\varphi$ ("φ holds eventually").

$\varphi_1 \mathsf{R} \varphi_2 := \neg(\neg\varphi_1 \mathsf{U} \neg\varphi_2)$.

Expressing properties of computations with LTL

(safety) $\texttt{init} \rightarrow \mathsf{G}\,\texttt{safe}$,

(liveness) $\mathsf{G}(\mathrm{p} \rightarrow \mathsf{F}\mathrm{q})$,

(total correctness) $(\texttt{init} \wedge \mathrm{p}) \rightarrow \mathsf{F}(\texttt{end} \wedge \mathrm{q})$,

(strong fairness) $\mathsf{GF}\,\texttt{enabled} \rightarrow \mathsf{GF}\,\texttt{executed}$.

Expressing properties with LTL: exercises

"Every occurrence of p will be followed immediately by an occurrence of q which will hold true until p ceases to be true."

$$G(p \rightarrow \mathsf{X}(q \wedge q\mathsf{U}\neg p)).$$

"Every time when a message is sent, it will not be marked as 'sent' before an acknowledgment of receipt is returned."

$$G(\mathrm{Sent} \rightarrow (\neg \mathrm{MarkedSent}\ \mathsf{U}\ \mathrm{AckReturned})).$$

"Between every two green signals there will be a red signal"

$$G(\mathit{Green} \rightarrow \mathsf{X}(\neg \mathit{Green}\ \mathsf{U}\ \mathit{Red}))?$$

or:

$$G(\mathit{Green} \rightarrow \mathsf{X}(\neg \mathit{Green}\ \mathsf{W}\ \mathit{Red}))?$$

or:

$$G(\mathit{Green} \wedge \mathsf{F}\,\mathit{Green} \rightarrow \mathsf{X}(\neg \mathit{Green}\ \mathsf{U}\ \mathit{Red}))?$$

Semantics of LTL

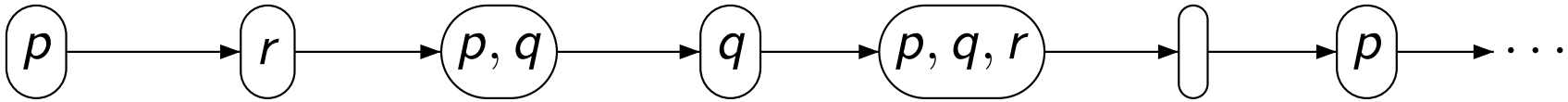
Given an LTL-model σ, a position $i \in \mathbb{N}$, and a formula φ, we define inductively the satisfaction relation $\models$ as follows:

- $\sigma, i \not\models \bot$,
- $\sigma, i \models \mathrm{p}$ iff $\mathrm{p} \in \sigma(i)$, for every $\mathrm{p} \in \mathrm{PROP}$,
- $\sigma, i \models \neg\varphi$ iff $\sigma, i \not\models \varphi$,
- $\sigma, i \models \varphi_1 \wedge \varphi_2$ iff $\sigma, i \models \varphi_1$ and $\sigma, i \models \varphi_2$,
- $\sigma, i \models \mathsf{X}\varphi$ iff $\sigma, i+1 \models \varphi$,
- $\sigma, i \models \mathsf{F}\varphi$ iff there is $j \geq i$ such that $\sigma, j \models \varphi$,
- $\sigma, i \models \mathsf{G}\varphi$ iff for all $j \geq i$, we have $\sigma, j \models \varphi$,
- $\sigma, i \models \varphi_1 \mathsf{U} \varphi_2$ iff there is $j \geq i$ such that $\sigma, j \models \varphi_2$ and $\sigma, k \models \varphi_1$ for all $i \leq k < j$.

We write $\sigma \models \varphi$ instead of $\sigma, 0 \models \varphi$.

An LTL-formula φ is satisfiable if there some LTL-model σ such that $\sigma \models \varphi$; φ is valid if $\neg\varphi$ is not satisfiable.

Semantics of LTL: exercises



The valuation defining the model σ:
$V(p) = \{s_{2k} \mid k \in \mathbb{N}\}$,
$V(q) = \{s_k \mid 2 \leq k \leq 4 \text{ or } 100 \leq k\}$,
$V(r) = \{s_{3k+1} \mid k \in \mathbb{N}\}$.

Determine:

$\sigma, 0 \stackrel{?}{\models} \mathsf{F}(q \wedge \mathsf{XX}\neg p)$ Yes; $\sigma, 0 \stackrel{?}{\models} \neg q \ \mathsf{U}\ (q \mathsf{U} r)$ Yes;

$\sigma \stackrel{?}{\models} \mathsf{FG}\neg(p \wedge q)$ No; $\sigma \stackrel{?}{\models} \mathsf{GF}\neg(p \wedge q)$ Yes;

$\sigma \stackrel{?}{\models} \mathsf{FGF}(p \wedge q \wedge r)$ Yes; $\sigma \stackrel{?}{\models} \mathsf{G}(p \rightarrow \mathsf{X}\neg p)$ Yes;

$\sigma, 1 \stackrel{?}{\models} \mathsf{F}(q \ \mathsf{U}\ \neg(p \vee q \vee r))$ Yes; $\sigma \stackrel{?}{\models} \mathsf{GF}((p \wedge \neg r)\mathsf{U} r)$ Yes;

$\sigma \stackrel{?}{\models} \mathsf{GF}(p \wedge \mathsf{G}q \wedge \mathsf{X}r)$; $\sigma \stackrel{?}{\models} \mathsf{GF}(r \ \mathsf{U}\ \mathsf{X}(\neg p \wedge \mathsf{X}r))$.

Some useful validities of LTL formulae

- $\mathsf{G}^{\infty}\varphi \rightarrow \mathsf{F}^{\infty}\varphi$
- $\mathsf{G}\varphi \wedge \mathsf{F}\psi \rightarrow \varphi \mathsf{U} \psi$
- $\varphi \wedge \mathsf{G}(\varphi \rightarrow \mathsf{X}\varphi) \rightarrow \mathsf{G}\varphi$
- $\varphi \wedge \mathsf{G}(\varphi \rightarrow \mathsf{XF}\varphi) \rightarrow \mathsf{GF}\varphi$
- $\mathsf{F}\varphi \leftrightarrow \varphi \vee \mathsf{XF}\varphi$
- $\mathsf{G}\varphi \leftrightarrow \varphi \wedge \mathsf{XG}\varphi$
- $\varphi \mathsf{U} \psi \leftrightarrow (\psi \vee (\varphi \wedge \mathsf{X}(\varphi \mathsf{U} \psi)))$

Truth of LTL-formulae in interpreted transition systems

Every rooted ITS generates a set of linear LTL-models, viz. all computations starting from the root.

So, LTL formulae can be evaluated in (rooted) ITS $(\mathcal{M}, s)$:

$(\mathcal{M}, s) \models \varphi$ iff $\sigma \models \varphi$ for all computations σ starting at s.

So, we define universal truth of φ at $(\mathcal{M}, s)$, denoted $\mathcal{M}, s \models_\forall \varphi$.

Likewise, we define existential truth of φ at $(\mathcal{M}, s)$, denoted $\mathcal{M}, s \models_\exists \varphi$.

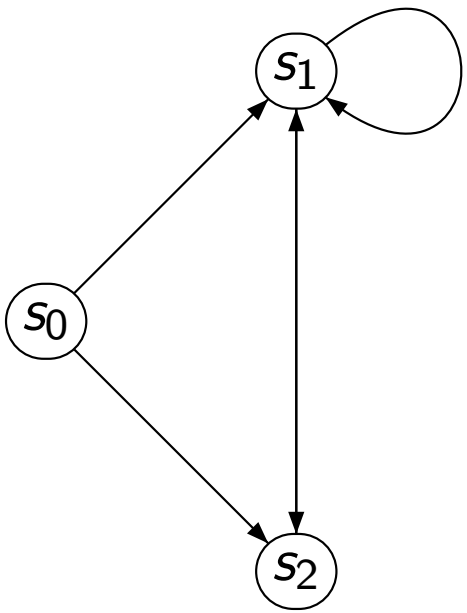
An ITS $\mathcal{M} = (S, R, L)$ is finite if each of S, the image of L, and each set in that image are finite sets.

A finite ITS can generate an uncountable set of computations.

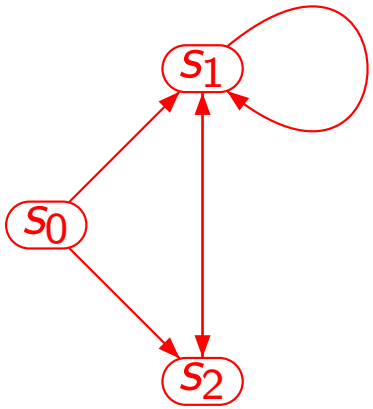
Unfolding of a transition system, intuitively

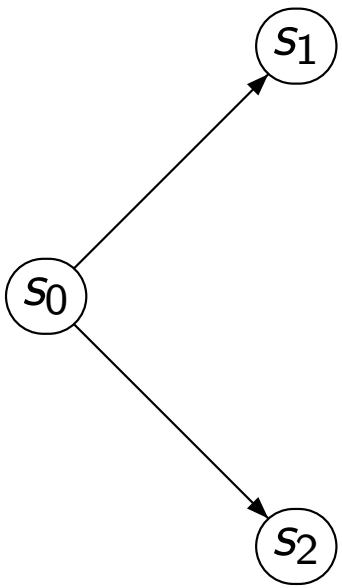
- Unfolding of a rooted transition system $(\mathcal{T}, s)$ starts from the root s, produces new copies of all successors of that state, and inserts respective transitions to these copies.
- Then the procedure continues recursively from each of these successor states.
- The result is a (usually infinite) tree in which all paths starting from the root of the original TS are presented explicitly as branches.

Unfolding of a rooted transition system 0

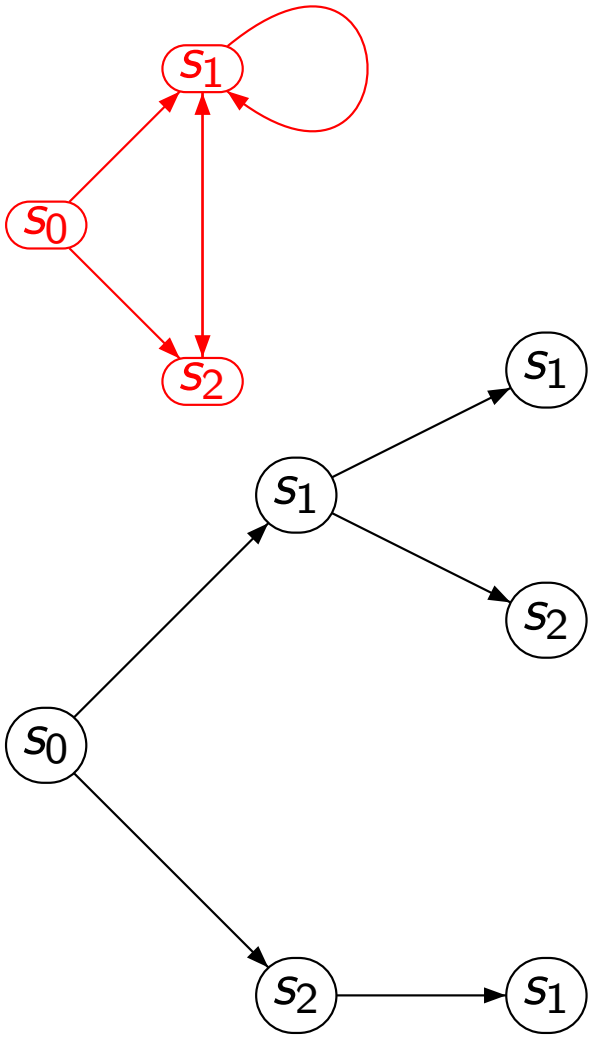


Unfolding of a rooted transition system 1

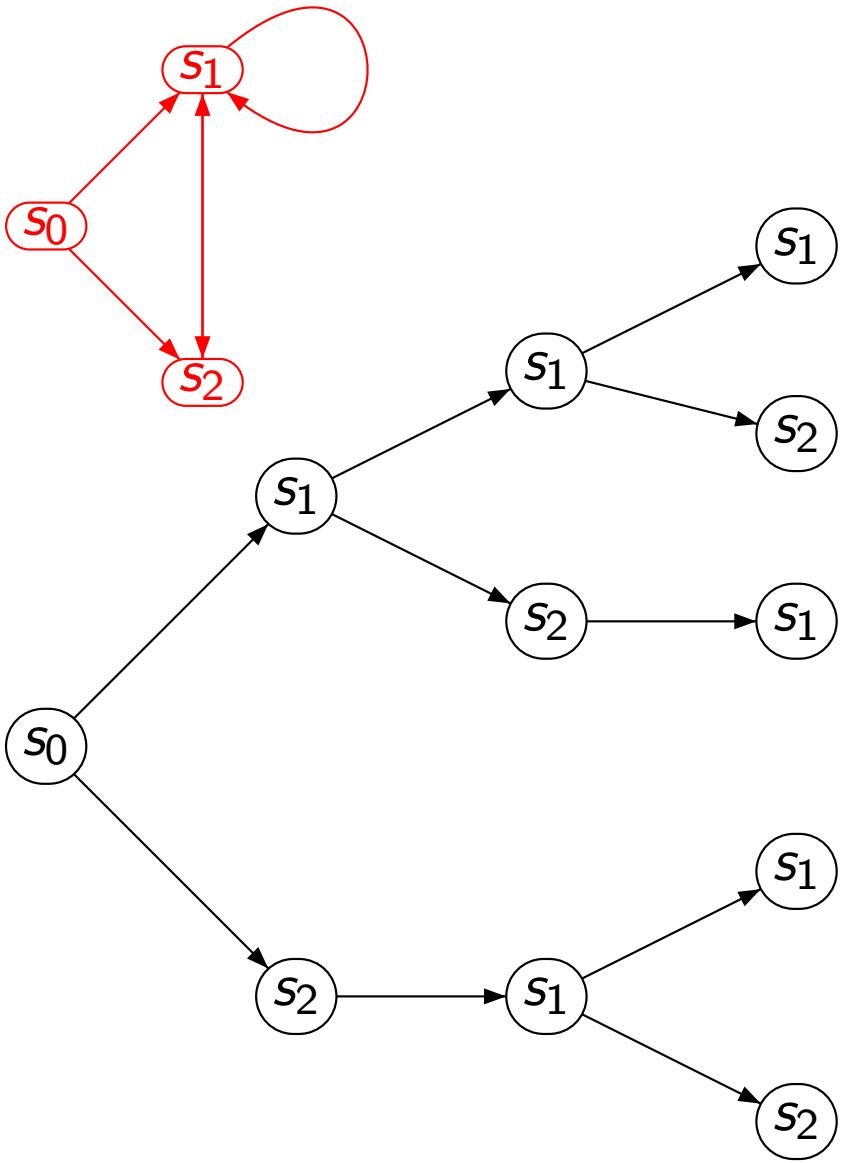




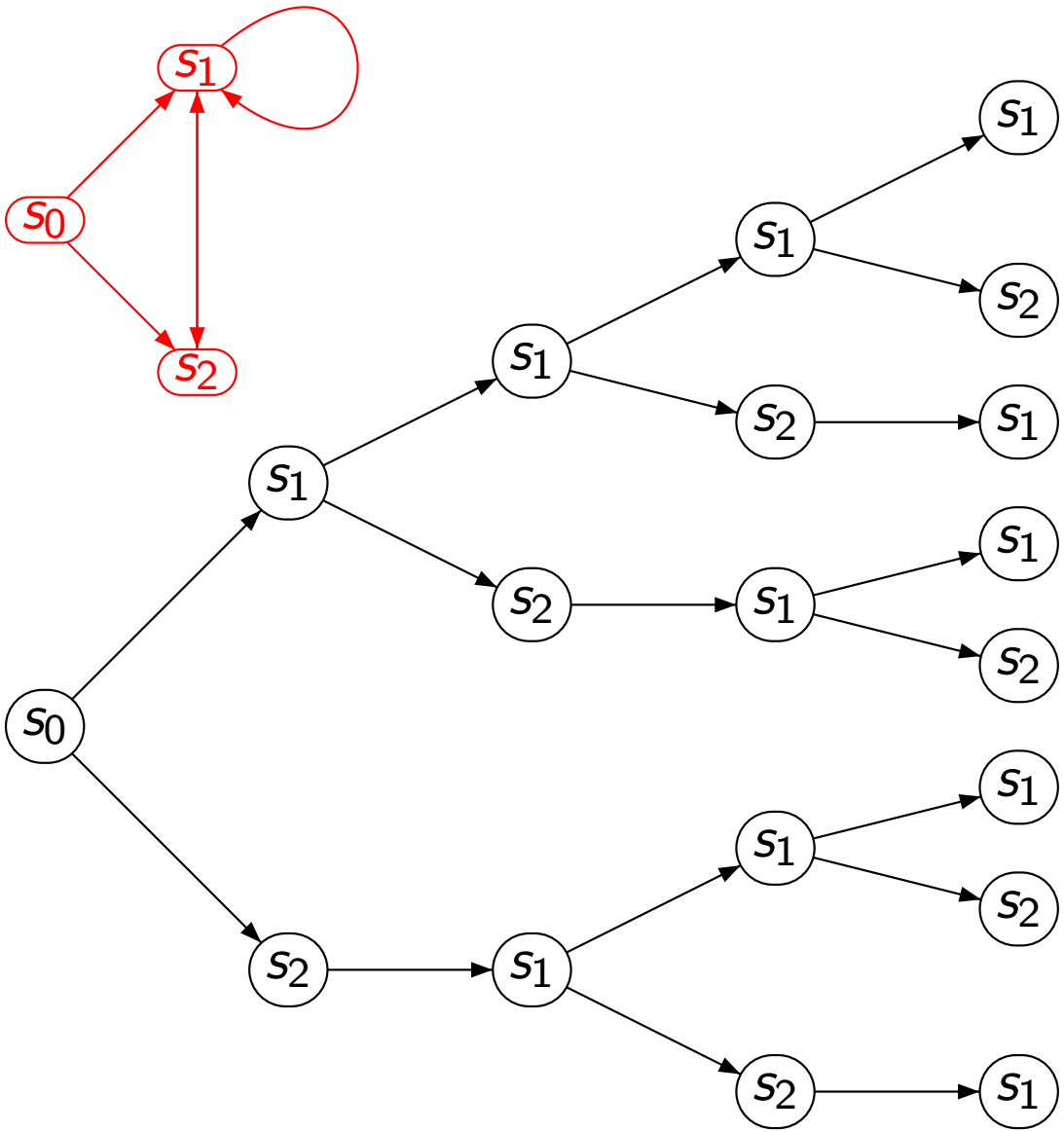
Unfolding of a rooted transition system 2



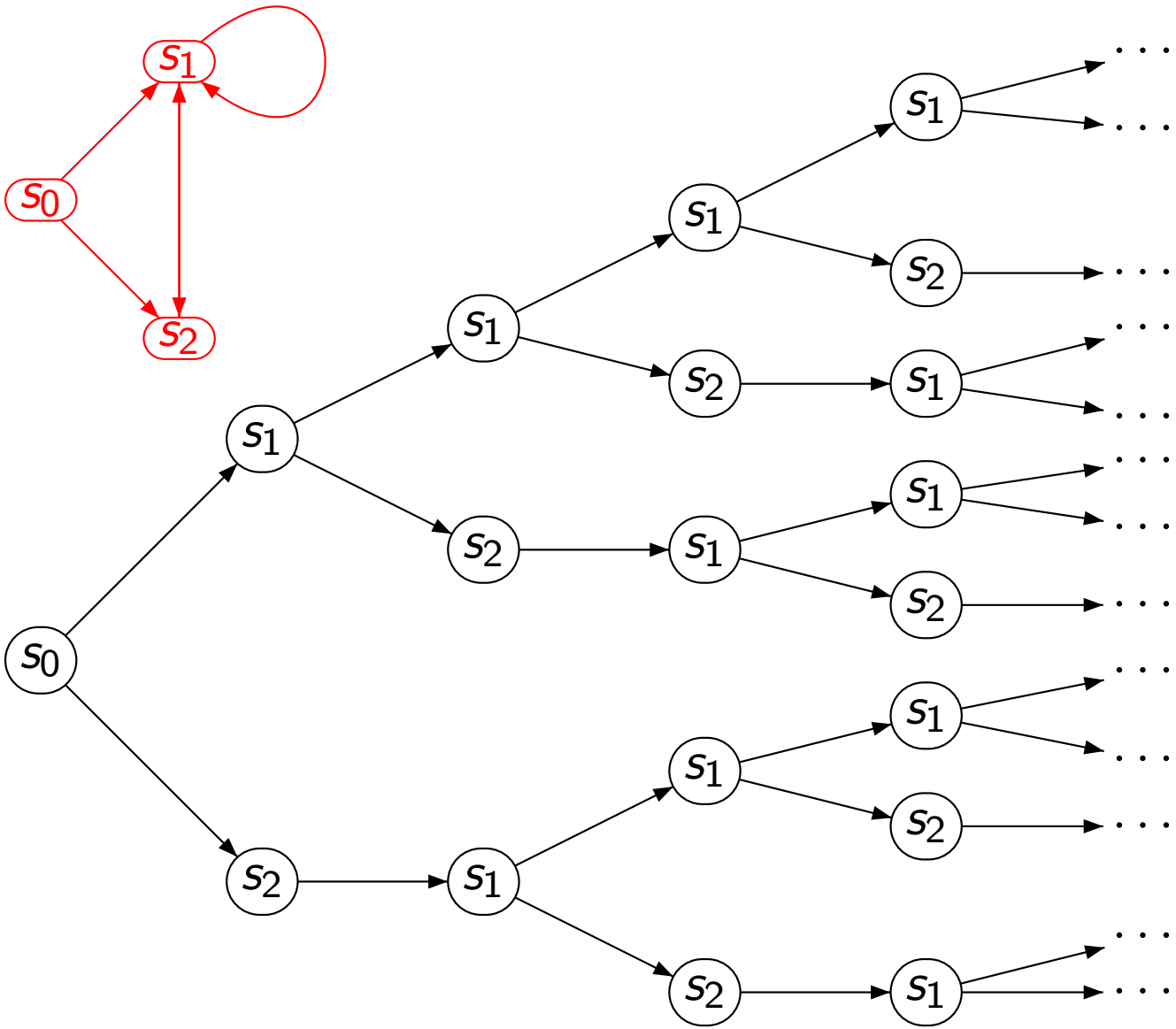
Unfolding of a rooted transition system 3



Unfolding of a rooted transition system 4



Unfolding of a rooted transition system 5



Preservation of truth of LTL-formulae in unfoldings

Let $\widehat{\mathcal{M}}, \widehat{s}$ be the unfolding of the ITS $\mathcal{M}$ from a state s in it. Then, paths and computations in $\mathcal{M}$ starting from s and in $\widehat{\mathcal{M}}$ starting from $\widehat{s}$ are in one-to-one correspondence, naturally mapping $\widehat{\mathcal{M}}$ onto $\mathcal{M}$: the last state of a path in $\widehat{\mathcal{M}}$ is actually the corresponding path in $\mathcal{M}$.

Consequently, for every LTL-formula φ:

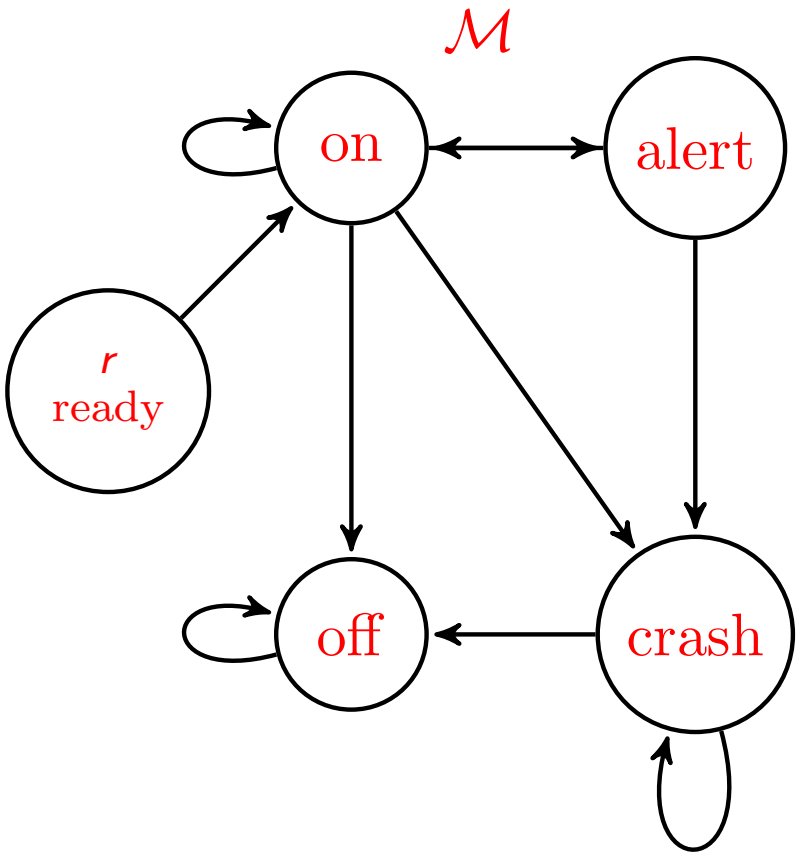
$$\mathcal{M}, s \models_\exists \varphi \text{ iff } \widehat{\mathcal{M}}, \widehat{s} \models_\exists \varphi$$

and likewise,

$$\mathcal{M}, s \models_\forall \varphi \text{ iff } \widehat{\mathcal{M}}, \widehat{s} \models_\forall \varphi.$$

The unfolding $(\widehat{\mathcal{M}}, \widehat{s})$ is also called the computation tree of $(\mathcal{M}, s)$.

Model checking of LTL-formulae in ITS: exercises



Check the following:

$\mathcal{M}, r \overset{?}{\models}_{\exists} \mathsf{G}\neg\ \mathrm{crash}$ Yes

$\mathcal{M}, r \overset{?}{\models}_{\forall} \mathsf{G}^{\infty}\neg\ \mathrm{crash}$ No

$\mathcal{M}, r \overset{?}{\models}_{\exists} \mathsf{F}^{\infty}\mathrm{alert} \wedge \mathsf{G}^{\infty}\mathrm{off}$

$\mathcal{M}, r \overset{?}{\models}_{\forall} \mathsf{G}^{\infty}(\mathrm{alert} \rightarrow \mathsf{F}\ \mathrm{crash})$

$\mathcal{M}, r \overset{?}{\models}_{\forall} \mathsf{F}^{\infty}(\mathrm{crash} \rightarrow \mathsf{X}\ \mathrm{off})$

$\mathcal{M}, r \overset{?}{\models}_{\exists} \mathsf{F}(\mathrm{on}\ \mathsf{U}\ \mathrm{off})$

$\mathcal{M}, r \overset{?}{\models}_{\forall} \mathsf{F}^{\infty}(\neg\mathrm{on}\ \mathsf{U}\ \mathrm{off})$

Ultimately periodic linear models for LTL

Ultimately periodic linear LTL-model, intuitively: like a lasso, consisting of a finite 'tail', followed by a finite 'loop'.

Formlly, a linear LTL-model $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ is ultimately periodic (UP) if there exist natural numbers i and $p > 0$ such that $\sigma(k) = \sigma(k + p)$ for every $k \geq i$.

The (possibly empty) sequence $\sigma(0), \ldots, \sigma(i-1)$ is the prefix and $\sigma(i), \ldots, \sigma(i+p-1)$ is the loop of the model.

We say that σ has prefix index i and period p.

Thus, an UP-model σ can be equivalently represented by the finite sequence $\Gamma_0, \ldots, \Gamma_{i+p}$ such that each $\Gamma_j = \sigma(j)$.

LTL model-checking as an algorithmic problem

In order to define the *algorithmic problem* of model-checking, models must be finitely presentable objects.

We say that $\mathcal{M} = (S, R, L)$ is finite whenever S, the image of L, and each set in that image are finite sets.

We only consider the algorithmic model-checking problem for LTL in finite ITS.

For instance, UP-models over finite sets of variables are not finite, but can be presented by finite ITS.

Path model-checking of LTL formulae

Model-checking for LTL in finite ITS is usually intractable.

If we restrict attention to finite UP-models, we have the path model-checking problem for LTL:

input: A finite UP-model σ and an LTL formula φ.

question: Does $\sigma \models \varphi$?

Proposition

The path model-checking problem for LTL is decidable in PTime.

Open question:

Is the path model-checking problem for LTL PTime-hard?

The full model-checking problems for LTL

Given a finite ITS $\mathcal{M} = (S, R, L)$, its size $|\mathcal{M}|$ is the sum

$$\mathrm{card}(S) + \mathrm{card}(R) + \sum_{s \in S} \mathrm{card}(L(s)).$$

The (universal) local model-checking problem for LTL:

input: an LTL formula φ, a finite ITS $\mathcal{M}$ and a state $s \in S$,

question: Is it the case that $\mathcal{M}, s \models_\forall \varphi$?.

In the above the codomain of L is restricted to $\mathrm{PROP}(\varphi)$.

Existential local model-checking, denoted by $\mathrm{MC}^\exists(\mathrm{LTL})$, is defined likewise.

Ultimately periodic model property for satisfiability of LTL-formulae

Theorem (Sistla& Clarke'85)

For every satisfiable LTL formula φ, there is an ultimately periodic model σ such that $\sigma \models \varphi$, its period is bounded by $|\varphi| \times 2^{4|\varphi|}$ and its prefix index is bounded by $2^{4|\varphi|}$.

Complexity of LTL

Theorem (Sistla& Clarke'85)

LTL satisfiability is in PSPACE.

Proof sketch: Guessing a sequence of length at most $2^{4|\varphi|} + |\varphi| \times 2^{4|\varphi|}$ and checking on-the-fly that it is a small satisfiability witness can be done in nondeterministic polynomial space.
By Savitch's Theorem this gives a polynomial space upper bound.

Theorem (Sistla& Clarke'85)

The existential and universal model-checking problems for LTL are in PSPACE.

Proof idea: By a similar argument, as it suffices to consider only the ultimately periodic computations starting from the given state. The only essential difference is that the maximal values for the prefix index and the period have an additional factor, viz., the number of states of the transition system.

Branching time temporal logics: intro

While LTL is suited for reasoning about single computations in a transition system, branching time logics are intended to reason about the *entire* transition system.

With formulae of branching time logics one can refer to, and quantify over, all possible computations starting from a given state.

The study and use of branching-time logics in computer science began in late 1970's - early 1980's, when several branching-time logics were proposed by Abrahamson, Lamport, Ben-Ari, Manna and Pnueli, Clarke and Emerson, Emerson and Halpern, etc.

Soon, a debate on the pros and cons of linear time vs branching time logics ensued.

The full computation tree logic CTL^*

In response to the debate on pros and cons of linear time vs branching time logics, Emerson and Halpern introduced in 1983 the very expressive logic CTL^*.

The language of CTL^* extends the one of LTL with the path quantifier A, were the formula $\mathsf{A}\,\varphi$ means "φ is true on *every* computation passing through the current state".

The set of formulae of CTL^* is defined recursively as follows:

$$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \mathsf{A}\,\varphi$$

The other logical connectives $\bot$, $\rightarrow$, $\vee$, $\leftrightarrow$, and the temporal operators F and G are definable as usual.

Macros: $\mathsf{F}^\infty := \mathsf{GF}$ and $\mathsf{G}^\infty := \mathsf{FG}$.

Existential path quantifier: $\mathsf{E}\,\varphi := \neg\,\mathsf{A}\,\neg\varphi$.

State and path formulae in CTL^*

Two types of CTL^*-formulae: state formulae $\mathrm{StateFor}$ and path formulae $\mathrm{PathFor}$, defined by mutual induction:

$\mathrm{StateFor}$:

- All atomic propositions and $\bot$ are in $\mathrm{StateFor}$.
- If $\varphi, \psi \in \mathrm{StateFor}$ then $\neg\varphi, \varphi \wedge \psi \in \mathrm{StateFor}$.
- If $\varphi \in \mathrm{PathFor}$, then $\mathsf{A}\,\varphi \in \mathrm{StateFor}$.

$\mathrm{PathFor}$:

- $\mathrm{StateFor} \subset \mathrm{PathFor}$.
- If $\varphi, \psi \in \mathrm{PathFor}$ then $\neg\varphi, \varphi \wedge \psi, \mathsf{X}\varphi, \varphi\mathsf{U}\psi \in \mathrm{PathFor}$.

Expressing properties of transition systems with CTL^*

- *Partial correctness along every possible computation:*

$$\varphi \rightarrow \mathsf{AG}(\mathrm{terminal} \rightarrow \psi).$$

- *Partial correctness along some possible computation:*

$$\varphi \rightarrow \mathsf{EG}(\mathrm{terminal} \rightarrow \psi).$$

- *Total correctness along every possible computation:*

$$\varphi \rightarrow \mathsf{AF}(\mathrm{terminal} \wedge \psi).$$

- and *total correctness along some possible computation:*

$$\varphi \rightarrow \mathsf{EF}(\mathrm{terminal} \wedge \psi).$$

- *Fairness along every possible computation:*

$$\mathsf{A}(\mathsf{GF}(\mathrm{Resource\ requested}) \rightarrow \mathsf{F}(\mathrm{Resource\ granted}))$$

Expressing properties with CTL^*: exercises

"On some computation starting from the current state every occurrence of p is eventually followed by an occurrence of q"

$$\mathsf{EG}(p \rightarrow \mathsf{F}p)$$

"From every state of every computation on which q is always true thereafter, a computation starts on which p will be true infinitely often".

$$\mathsf{AG}(\mathsf{G}q \rightarrow \mathsf{EGF}p)$$

"There is no computation on which some state where q is true starts a computation on which p is never true."

$$\neg\, \mathsf{EF}(q \wedge \mathsf{EG}\neg p)$$

"On every computation the successor state of some state where p is true starts a computation on which q is true until p becomes false."

$$\mathsf{AF}(p \wedge \mathsf{X}\,\mathsf{E}(q\mathsf{U}\neg p)).$$

Formal semantics of CTL^*

CTL^*-models: interpreted transition systems

Let $\mathcal{M} = \langle S, R, L \rangle$ is an ITS and π is a path in $\mathcal{M}$.
$\mathcal{M}, \pi \models \varphi$ means that φ is true on the path π in $\mathcal{M}$.

Notation: given a path π, we obtain the path $\pi_{\geq k}$ by chopping off the first k states of π, i.e., $\pi_{\geq k} = \pi(k), \pi(k+1), \pi(k+2), \ldots$.

The inductive definition of $\mathcal{M}, \pi \models \varphi$:

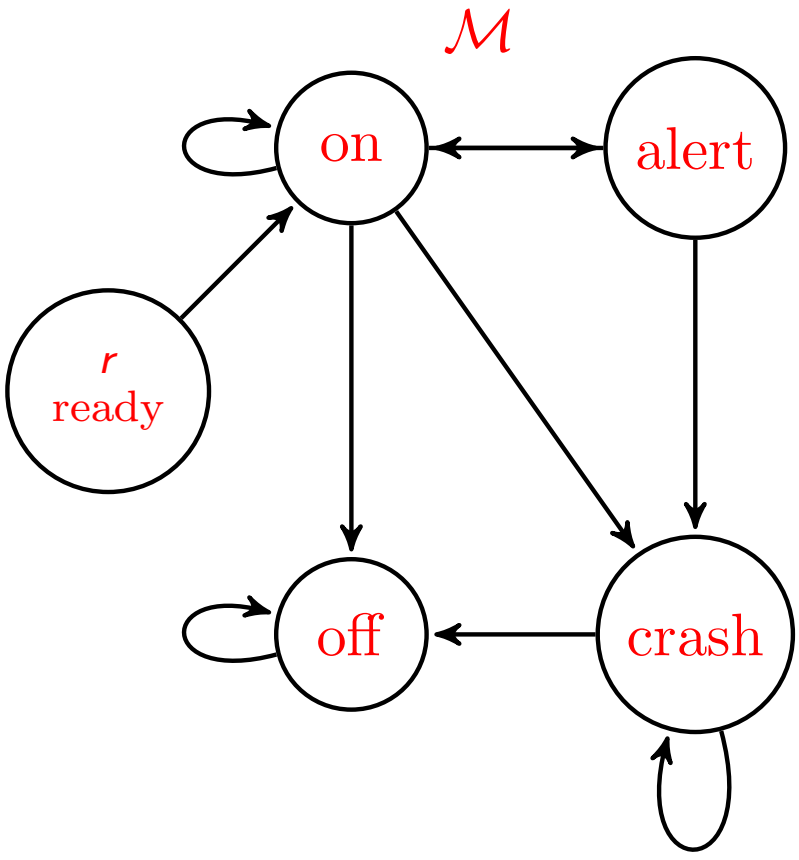
- $\mathcal{M}, \pi \models p$ iff $p \in L(\pi(0))$ for $p \in \mathrm{PROP}$;
- $\mathcal{M}, \pi \models \neg\varphi$ iff $\mathcal{M}, \pi \not\models \varphi$;
- $\mathcal{M}, \pi \models \varphi \wedge \psi$ iff $\mathcal{M}, \pi \models \varphi$ and $\mathcal{M}, \pi \models \psi$;
- $\mathcal{M}, \pi \models \mathsf{X}\varphi$ iff $\mathcal{M}, \pi_{\geq 1} \models \varphi$;
- $\mathcal{M}, \pi \models \varphi \mathsf{U} \psi$, iff $\mathcal{M}, \pi_{\geq j} \models \psi$ for some $j \geq 0$ and $\mathcal{M}, \pi_{\geq i} \models \varphi$ for every i such that $0 \leq i < j$.
- $\mathcal{M}, \pi \models \mathsf{A}\varphi$ iff $\mathcal{M}, \pi' \models \varphi$ for every path π' in $\mathcal{M}$ with the same initial state as π.

Preservation of truth of CTL^*-formulae in unfoldings

For every state formula φ of CTL^*, every interpreted transition system $\mathcal{M}$, and a state s in it:

$$\mathcal{M}, s \models \varphi \text{ iff } \widehat{\mathcal{M}}, \widehat{s} \models \varphi$$

Model checking of CTL^*-formulae in ITS: exercises



Check the following:

$\mathcal{M}, r \stackrel{?}{\models} \mathsf{EG}\neg\ \mathrm{crash}$ Yes

$\mathcal{M}, r \stackrel{?}{\models} \mathsf{AGF}\neg\ \mathrm{crash}$ No

$\mathcal{M}, r \stackrel{?}{\models} \mathsf{AG}\,\mathsf{EF}\neg\ \mathrm{crash}$ Yes

$\mathcal{M}, r \stackrel{?}{\models} \mathsf{A}(\mathsf{GF}\mathrm{alert} \rightarrow \mathsf{GF}(\mathrm{on} \vee \mathrm{crash}))$

$\mathcal{M}, r \stackrel{?}{\models} \mathsf{EG}(\mathsf{EX}\ \mathrm{crash} \wedge \mathsf{EG}\neg\mathrm{crash})$

$\mathcal{M}, r \stackrel{?}{\models} \mathsf{AG}\,\mathsf{EF}(\mathrm{on}\ \mathsf{U}\ \mathrm{off})$

Truth, validity, satisfiability

Given a CTL^*-model $\mathcal{M}$:

- path formula φ is true of the state s of $\mathcal{M}$, denoted $\mathcal{M}, s \models \varphi$, if $\mathcal{M}, \pi \models \varphi$ for every path π in $\mathcal{M}$ such that $\pi(0) = s$.
- formula φ is valid in $\mathcal{M}$, denoted $\mathcal{M} \models \varphi$, if $\mathcal{M}, s \models \varphi$ for every state s in $\mathcal{M}$.
- formula φ is valid, denoted $\models \varphi$, if it is valid in every ITS.
- path formula φ is satisfiable if it is true of some path π in some interpreted transition system $\mathcal{M}$.
- likewise, a state formula φ is satisfiable if it is true of some state s in some interpreted transition system $\mathcal{M}$.

Note that validity in CTL^* is not closed under uniform substitutions.
Indeed, $p \to \mathsf{A}\, p$ is valid for any $p \in \mathrm{PROP}$, while $\mathsf{G}p \to \mathsf{AG}p$ is not valid.

Some useful validities in CTL^*

The following formulae are CTL^*-valid:

- $\mathsf{A}\,\varphi$ for every LTL-valid formula φ;
- $\mathsf{A}\,\varphi \to \varphi$; (NB: φ can be a path or a state formula.)
- $\mathsf{A}(\varphi \to \psi) \to (\mathsf{A}\,\varphi \to \mathsf{A}\,\psi)$;
- $\mathsf{AX}\varphi \to \mathsf{X}\,\mathsf{A}\,\varphi$; $\quad \mathsf{A}\,\mathsf{G}\varphi \to \mathsf{G}\,\mathsf{A}\,\varphi$;
- $\mathsf{AG}(\varphi \to \mathsf{EX}\varphi) \to (\varphi \to \mathsf{EG}\varphi)$;
- $\mathsf{AG}\varphi \leftrightarrow \varphi \wedge \mathsf{AX}\,\mathsf{AG}\varphi$; $\quad \mathsf{EG}\varphi \leftrightarrow \varphi \wedge \mathsf{EX}\,\mathsf{EG}\varphi$;
- $\mathsf{A}(\varphi \mathsf{U} \psi) \leftrightarrow \psi \vee \mathsf{AX}\,\mathsf{A}(\varphi \mathsf{U} \psi)$; $\quad \mathsf{E}(\varphi \mathsf{U} \psi) \leftrightarrow \psi \vee \mathsf{EX}\,\mathsf{E}(\varphi \mathsf{U} \psi)$;
- $\mathsf{A}\,\mathsf{G}\,\mathsf{EF}\varphi \to \mathsf{E}\,\mathsf{GF}\varphi$; (Burgess's formula)
- $\mathsf{A}\,\mathsf{G}(\mathsf{A}\,\varphi \to \mathsf{EXF}\,\mathsf{A}\,\varphi) \to (\mathsf{A}\,\varphi \to \exists\mathsf{GF}\,\mathsf{A}\,\varphi)$;
- $\mathsf{A}\,\mathsf{G}(\mathsf{E}\,\varphi \to \mathsf{E}\,\mathsf{X}((\mathsf{E}\,\psi \mathsf{U}\,\mathsf{E}\,\theta))) \to (\mathsf{E}\,\varphi \to \mathsf{E}\,\mathsf{G}((\mathsf{E}\,\psi \mathsf{U}\,\mathsf{E}\,\theta)))$ (Reynold's limit closure formula).

CTL as a fragment of CTL^*

The computation tree logic CTL: fragment of CTL^*, introduced in 1980-81, before CTL^*, by Clarke and Emerson.

The language and syntax of CTL are the same as those of CTL^*. Syntactic restriction on the CTL-formulae: the temporal operators must be immediately quantified by path quantifiers.

Thus, in CTL there are only state formulae.

For instance $\mathsf{A}\,\mathsf{GF}\varphi$ and $\mathsf{E}(\mathsf{F}\varphi \wedge \varphi \mathsf{U} \psi)$ are not CTL-formulae. $\mathsf{A}(\varphi_1 \mathsf{U} \varphi_2)$ and $\mathsf{E}(\varphi_1 \mathsf{U} \varphi_2)$ are not inter-definable in CTL, so both path quantifiers must be present in the language.

Recursive definition of CTL-formulae:

$$\varphi := \top \mid p \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \mathsf{AX}\varphi \mid \mathsf{A}(\varphi_1 \mathsf{U} \varphi_2) \mid \mathsf{E}(\varphi_1 \mathsf{U} \varphi_2).$$

The semantics of CTL is the same as CTL^*.

Some definable operators in CTL

- $\mathsf{EX}\varphi := \neg\,\mathsf{AX}\neg\varphi$,
- $\mathsf{AF}\varphi := \mathsf{A}(\top\mathsf{U}\varphi)$,
- $\mathsf{EFF}\varphi := \mathsf{E}(\top\mathsf{U}\varphi)$,
- $\mathsf{AG}\varphi := \neg\,\mathsf{EF}\neg\varphi$,
- $\mathsf{EG}\varphi := \neg\,\mathsf{AF}\neg\varphi$.

Thus, LTL can be regarded as the fragment of CTL^* consisting of all *purely path formulae*, i.e., those containing no path quantifiers, while CTL is the fragment of CTL^* consisting of all *purely state formulae*.

Expressiveness and extensions of CTL

For invariance and eventuality properties, CTL is essentially as expressive as CTL^*. However, CTL is not suitable for expressing *fairness properties* where G^∞ and F^∞ are essentially used.

(The original version of CTL contained these operators, too.)

E.g., fairness along every possible computation, expressible in CTL^* is beyond the expressiveness of CTL.

But, how about:

$$\mathsf{AG}\,\mathsf{AF}(\text{resource requested}) \rightarrow \mathsf{AF}(\text{resource granted})?$$

Some extensions of CTL:

- CTL^2: allows pairing of two temporal operators after a path quantifier, thus enabling expression of fairness properties.
- CTL^+: allows any boolean combinations of unnested temporal operators in the scope of a path quantifier.
  As expressive as CTL, but exponentially more succinct.
- CTL with past.

Model-checking of CTL^* reduced to model-checking of LTL

Model-checking of CTL^*-formulae can be reduced to global model-checking of LTL-formulae.

The idea:

- CTL^*-formulae with path quantifier rank 0 are precisely the LTL-formulae.
- For a CTL^*-formula of a positive path quantifier rank replace the maximal state subformulae by fresh propositional variables to obtain an LTL-formula. Compute the extensions of the maximal state subformulae recursively.

This procedure gives a PSPACE-complete method for model-checking of CTL^* (optimal since LTL is a fragment of CTL^*).

Example of model checking CTL^*-formula

Given: CTL^*-model $\mathcal{M}$, a computation σ in it, and the CTL^*-formula $\alpha = \mathsf{G}(\mathsf{A}\,\varphi \to \mathsf{X}\,\mathsf{EF}\varphi) \wedge \mathsf{A}\,\varphi \wedge \neg\mathsf{G}\,\mathsf{EF}\varphi$ for some formula φ.

To determine whether $\mathcal{M}, \sigma \models \alpha$ do the following.

The maximal state subformulae of α are $\mathsf{A}\,\varphi$ and $\mathsf{EF}\varphi$.

Replace these respectively by p and q (both non-occurring in φ) to obtain the LTL-formula $\alpha' = \mathsf{G}(p \to \mathsf{X}q) \wedge p \wedge \neg\mathsf{G}q$.

Applying recursively the same procedure, do global model-checking of each of $\mathsf{A}\,\varphi$ and $\mathsf{EF}\varphi$ to compute $\|\mathsf{A}\,\varphi\|$ and $\|\mathsf{EF}\varphi\|$ in $\mathcal{M}$.

Then, modify $\mathcal{M}$ into $\mathcal{M}'$ by re-defining L to L', so that $\|p\| = \|\mathsf{A}\,\varphi\|$ and $\|q\| = \|\mathsf{EF}\varphi\|$.

We claim that $\mathcal{M}, \sigma \models \alpha$ iff $\mathcal{M}', \sigma \models \alpha'$.

Model-checking problems for CTL

The local CTL model-checking problem LMC(CTL):

input: a CTL formula φ, a finite ITS $\mathcal{M}$, and $s_0 \in \mathcal{M}$.

output: 1 if $\mathcal{M}, s_0 \models \varphi$, 0 otherwise.

The global model-checking problem GMC(CTL):

input: a CTL formula φ and a finite ITS $\mathcal{M}$.

output: the set $\|\varphi\|_{\mathcal{M}} = \{s \in \mathcal{M} : \mathcal{M}, s \models \varphi\}$.

Polynomial time model-checking algorithm for CTL

Unlike CTL^*, model-checking of CTL-formulae can be done very efficiently.

Proposition (Clarke and Emerson, 1981)

Let $\mathcal{M} = (W, R, L)$ be a CTL model and φ be a CTL formula. $\|\varphi\|_{\mathcal{M}}$ can be computed in time $\mathcal{O}((\mathrm{card}(R) + \mathrm{card}(W)) \times |\varphi|)$.

PTIME model-checking algorithm for CTL: sketch

Given an ITS $\mathcal{M}$ and a formula θ, the algorithm computes the extension $\|\theta\|_{\mathcal{M}}$ inductively on the structure of θ.

The propositional cases are routine.

$\|\,\mathsf{EX}\varphi\|$ consists of all states which have a successor in $\|\varphi\|$.

$\|\,\mathsf{AX}\varphi\|$ consists of all states which have all their successors in $\|\varphi\|$.

$\|\,\mathsf{E}\,\varphi\mathsf{U}\psi\|$ consists of all states from which $\|\psi\|$ is reachable by a path within $\|\varphi\|$. These are computed iteratively, in linear time.

$\|\,\mathsf{EG}\varphi\|$ consists of all states from which some strongly connected component (SCC) in $\|\varphi\|$ is reachable by a path within $\|\varphi\|$. Again, these are computed in linear time.

Exercise: work out the case $\|\,\mathsf{A}\,\varphi\mathsf{U}\psi\|$.

Exercise: extend the algorithm to G^{∞}.

Decidability and complexity of satisfiability testing for CTL* and CTL

Theorem (Clarke & Emerson, 1981)

The satisfiability problem for CTL *is* ExpTime*-complete.*

Proof sketch: The upper bound is provided by the tableau construction that will be described later.
The matching lower bound is by reduction from alternating polynomial space bounded Turing machines.

Theorem (Vardi & Stockmeyer, 1985; Emerson &Jutla, 1999)

The satisfiability problem for CTL* *is* 2ExpTime*-complete.*

Proof sketch: The lower bound is established by Vardi and Stockmeyer by reduction from alternating exponential space bounded Turing machines.
The upper bound is proved by Emerson and Jutla by an elaborated reduction to non-emptiness of automata on infinite trees.

Linear-time vs branching-time logics

Debate on pros and cons of linear vs branching time temporal logics in terms of:

- *expressiveness*,
- *complexity of satisfiability*,
- *complexity of model checking*,
- *suitability for specific practical purposes*.

Linear-time vs branching-time logics: expressiveness

LTL and CTL are expressively incompatible:

1. The CTL-formula AF AGp is not expressible in LTL.
2. The LTL-formula A FGp is not expressible in CTL.

Proof.
Every CTL^* formula φ is associated with the LTL-formula $\mathrm{LTL}(\varphi)$ obtained from φ by deleting all path quantifiers.

Now, for every path π in a CTL^*-model $\mathcal{M}$, we have that $\pi \models \mathrm{LTL}(\varphi)$ iff $\mathcal{M}_\pi, \pi \vDash \varphi$, where $\mathcal{M}_\pi$ is the CTL^*-model obtained by restricting $\mathcal{M}$ over the path π.

Exercise: complete the proof. □

CTL^* subsumes both rivals but is computationally expensive.

Linear-time vs branching-time logics: complexity

CTL*: satisfiability testing is deterministic 2ExpTime-complete; model-checking is PSpace-complete. Both are intractable.

CTL: good for model-checking (bilinear time complexity) but bad for satisfiability testing (deterministic ExpTime-complete)

LTL: worse for model-checking, but better for satisfiability testing (both PSpace-complete).

Summary

Each of linear-time and branching-time logics has its pros and cons as a framework for formal specification.

They are generally incompatible and there are no decisive theoretical arguments in favour of one approach to the other.

Vardi: "*The debate might end up being decided by the market-place rather than by the research community*".