EECS 144/244: Fundamental Algorithms for System Modeling, Analysis, and Optimization

Discrete Systems

Lecture: Transition systems, Temporal logic

Stavros Tripakis

University of California, Berkeley

TRANSITION SYSTEMS

Transition Systems

An even more basic model than automata and state machines:

transition system $=$ states $+$ transitions ($+$ labels)

But: possibly infinite sets of states/transitions.

- Can describe infinite-state systems (e.g., software).
- Can also be used in non-discrete systems (e.g., timed automata or dataflow graphs, as we will see later).
- Form the basis for the semantics of temporal logics (later in this lecture) and other equivalences between systems (e.g., bisimulation).

Many variants: Labeled Transition Systems, Kripke Structures, ...

Labeled Transition Systems

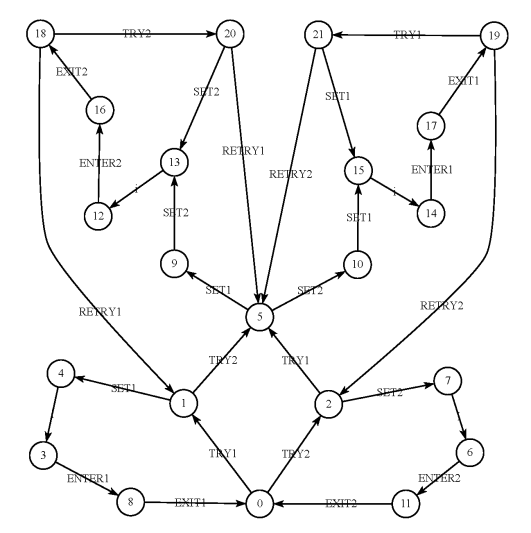
An LTS is a tuple:

$$(\Sigma, S, S_0, R)$$

- Σ: set of labels (modeling events, actions, ...)
- S: set of states (perhaps infinite)
- $S_0 \subseteq S$: set of initial states
- R: transition relation

$$R \subseteq S \times \Sigma \times S$$

Example: LTS


18
TRY2
20
21
TRY1
19
EXIT2
SET2
SET1
EXIT1
16
17
13
RETRY1
RETRY2
15
ENTER2
ENTER1
i
12
SET2
SET1
i
14
9
10
SET1
SET2
5
RETRY1
RETRY2
4
TRY2
TRY1
7
SET1
1
2
SET2
3
TRY1
TRY2
6
ENTER1
ENTER2
8
EXIT1
0
EXIT2
11

Kripke Structures

A Kripke structure is a tuple:

$$(P, S, S_0, L, R)$$

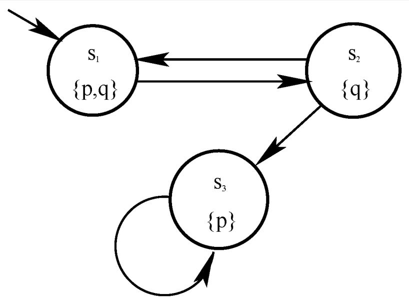
- P: set of atomic propositions (modeling state properties)
- S: set of states (perhaps infinite)
- $S_0 \subseteq S$: set of initial states
- L: labeling function on states

$$L : S \to 2^P$$

  For $p \in P$ and $s \in S$: "s has property p" iff $p \in L(s)$.
- R: transition relation

$$R \subseteq S \times S$$

Example: Kripke Structure



(Linear) Paths, Traces, ...

A *path* in a Kripke structure (P, S, S_0, L, R) is a (finite or infinite) sequence of states:

$$s_0, s_1, s_2, \cdots$$

such that

- $s_0 \in S_0$
- $\forall i : (s_i, s_{i+1}) \in R$

(Linear) Paths, Traces, ...

A *path* in a Kripke structure (P, S, S_0, L, R) is a (finite or infinite) sequence of states:

$$s_0, s_1, s_2, \cdots$$

such that

- $s_0 \in S_0$
- $\forall i : (s_i, s_{i+1}) \in R$

An observable *trace* σ is the corresponding sequence of sets of properties:

$$\sigma = L(s_0), L(s_1), L(s_2), \cdots$$

(Linear) Paths, Traces, ...

A *path* in a Kripke structure (P, S, S_0, L, R) is a (finite or infinite) sequence of states:

$$s_0, s_1, s_2, \cdots$$

such that

- $s_0 \in S_0$
- $\forall i : (s_i, s_{i+1}) \in R$

An observable *trace* σ is the corresponding sequence of sets of properties:

$$\sigma = L(s_0), L(s_1), L(s_2), \cdots$$

Note: we can look at a trace also as a sequence of *minterms*.
Example: if $P = \{p, q\}$ then a possible trace is:

$$\sigma = \{p\}, \{q\}, \{p, q\}, \{\}, \{p, q\}, ... = p\overline{q}, \overline{p}q, pq, \overline{pq}, pq, ...$$

Homework: Can we translate a KS to an "equivalent" LTS (and what does equivalent mean)? And vice-versa?

Reachable State Space

Given Kripke structure (P, S, S_0, L, R), a state $s \in S$ is called *reachable* if there exists a finite path:

$$s_0, s_1, s_2, \cdots, s_n$$

such that $s_n = s$.

Reachable state space: the sub-graph containing only reachable states and their transitions.

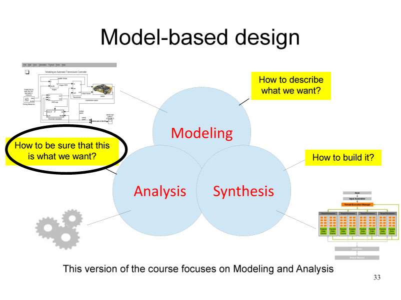
Deadlocks

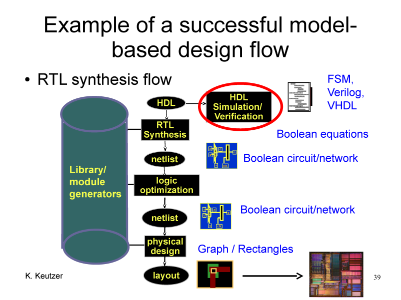
A state s is a *deadlock* if it has no outgoing transitions:

$$\nexists s' : (s, s') \in R$$

TEMPORAL LOGIC

Why temporal logic



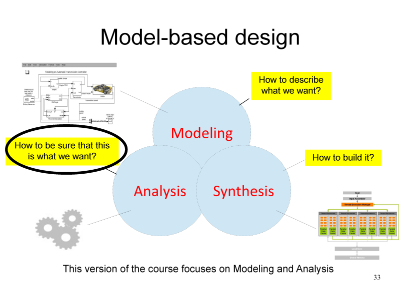


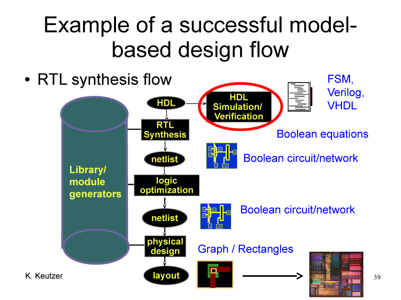
a formal specification language

=

a way to specify what we want mathematically
(and unambiguously!)

Why temporal logic





a formal specification language

=

a way to specify what we want mathematically

(and unambiguously!)

Amir Pnueli (1941 - 2009) won the ACM Turing Award, in part for proposing to use temporal logic for specification.

Example: Specification of the SpaceWire Protocol (European Space Agency standard)

8.5.2.2 ErrorReset

a. The *ErrorReset* state shall be entered after a system reset, after link operation is terminated for any reason or if there is an error during link initialization.

b. In the *ErrorReset* state the Transmitter and Receiver shall all be reset.

c. When the reset signal is de-asserted the *ErrorReset* state shall be left unconditionally after a delay of 6,4 µs (nominal) and the state machine shall move to the *ErrorWait* state.

d. Whenever the reset signal is asserted the state machine shall move immediately to the *ErrorReset* state and remain there until the reset signal is de-asserted.

From Sanjit Seshia.

Current status of property specification in HW

Usage of formal property specification languages is becoming widespread

- **68% in 2007 (John Cooley, DVCon'07)**
- **Properties often called "assertions"**

Properties are used not just in formal verification, but also in simulation

- **"Assertion-Based Verification" (ABV)**

Some property specification languages: PSL/Sugar, System Verilog Assertions (SVA), OVA, OVL, etc.

All of these are just ways of writing variants of Temporal Logic

6

From Sanjit Seshia.

Assertion-Based Verification

Monitor generated from assertions:

Temporal Logics

- Many variants: for linear, branching, timed, continuous, security, ..., properties
- We will look at LTL (linear) and CTL (branching).

LTL (Linear Temporal Logic) – Syntax

LTL[1] formulas are defined by the following grammar:

$$
\begin{array}{rcl}
\phi & ::= & p \mid q \mid \ldots \\
 & & \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \\
 & & \mid G\phi_1 \\
 & & \mid F\phi_1 \\
 & & \mid X\phi_1 \\
 & & \mid \phi_1 U \phi_2
\end{array}
$$

$G\phi$ is also written $\Box\phi$.
$F\phi$ is also written $\Diamond\phi$.
$X\phi$ is also written $\bigcirc\phi$.

[1] In fact this is PLTL: Propositional LTL (vs. first-order LTL).

LTL (Linear Temporal Logic) – Syntax

Examples:

$$GFp$$

$$G(p \rightarrow Fq)$$

$$pU(qU(p \wedge r))$$

LTL (Linear Temporal Logic) – Syntax

Examples:

$$GFp$$

$$G(p \rightarrow Fq)$$

$$pU(qU(p \wedge r))$$

What do these formulas mean?

LTL – Semantics: Intuition

LTL formulas are evaluated over infinite sequences (traces).
Satisfaction relation looks like this – for LTL formula ϕ and infinite trace σ:

$$\sigma \models \phi$$

formula	mnemonic
p	proposition (must hold now)
$\mathbf{G}\phi$	always, globally
$\mathbf{F}\phi$	finally, future, eventually
$\mathbf{X}\phi$	next step
$\phi_1 \mathbf{U} \phi_2$	until

Intuitive semantics

LTL – Semantics: Formally

We want to define formally the satisfaction relation: $\sigma \models \phi$.

What kind of object is σ ?

An infinite trace, usually coming from a Kripke structure (P, S, S_0, L, R):

$$\sigma = \sigma_0, \sigma_1, \sigma_2, \cdots$$

where $\sigma_i \subseteq P$ for all i.

Example:

$$\sigma = \{p\}, \{q\}, \{p, q\}, \{\}, \{p, q\}, ...$$

LTL – Semantics: Formally

Let

$$\sigma = \sigma_0, \sigma_1, \sigma_2, \cdots$$

Notation: $\sigma[i..] = \sigma_i, \sigma_{i+1}, \sigma_{i+2}, \cdots$

Satisfaction relation defined recursively on the syntax of a formula:

$$
\begin{array}{lll}
\sigma \models p & \text{iff} & p \in \sigma_0 \\
\sigma \models \phi_1 \wedge \phi_2 & \text{iff} & \sigma \models \phi_1 \text{ and } \sigma \models \phi_2 \\
\sigma \models \neg\phi & \text{iff} & \sigma \not\models \phi \\
\sigma \models G\phi & \text{iff} & \forall i = 0, 1, \ldots : \sigma[i..] \models \phi \\
\sigma \models F\phi & \text{iff} & \exists i = 0, 1, \ldots : \sigma[i..] \models \phi \\
\sigma \models X\phi & \text{iff} & \sigma[1..] \models \phi \\
\sigma \models \phi_1 U \phi_2 & \text{iff} & \exists i = 0, 1, \ldots : \sigma[i..] \models \phi_2 \wedge \\
 & & \qquad \forall 0 \leq j < i : \sigma[j..] \models \phi_1
\end{array}
$$

LTL – Semantics: Formally

Let

$$\sigma = \sigma_0, \sigma_1, \sigma_2, \cdots$$

Notation: $\sigma[i..] = \sigma_i, \sigma_{i+1}, \sigma_{i+2}, \cdots$

Satisfaction relation defined recursively on the syntax of a formula:

$\sigma \models p$ iff $p \in \sigma_0$ — p holds at the first (current) step

$\sigma \models \phi_1 \wedge \phi_2$ iff $\sigma \models \phi_1$ and $\sigma \models \phi_2$

$\sigma \models \neg\phi$ iff $\sigma \not\models \phi$

$\sigma \models G\phi$ iff $\forall i = 0, 1, ... : \sigma[i..] \models \phi$ — ϕ holds for every suffix of σ

$\sigma \models F\phi$ iff $\exists i = 0, 1, ... : \sigma[i..] \models \phi$ — ϕ holds for some suffix of σ

$\sigma \models X\phi$ iff $\sigma[1..] \models \phi$ — ϕ holds for the suffix starting at the next step

$\sigma \models \phi_1 U \phi_2$ iff $\exists i = 0, 1, ... : \sigma[i..] \models \phi_2 \wedge \forall 0 \leq j < i : \sigma[j..] \models \phi_1$

ϕ_2 holds for some suffix of σ and
ϕ_1 holds for all previous suffixes

LTL – examples

See `http://embedded.eecs.berkeley.edu/eecsx44/fall2011/lectures/TemporalLogic.pdf`.

Errata:

- Slides 13-14: "if and only if it holds" should be "if and only if p holds".
- Slide 19: $F(p \Rightarrow (XXq))$ should be $G(p \Rightarrow (XXq))$.

Linear-Time vs. Branching-Time Semantics

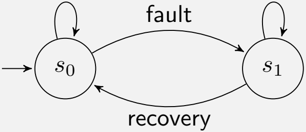
Some properties cannot be expressed on linear behaviors, e.g.:

> *"it is possible to recover from any fault"*

or

> *"there exists a way to get back to the initial state from any reachable state"*

Based on *one* (linear) behavior alone,[2] we cannot conclude whether our system satisfies the property.
E.g., the following system satisfies the property, although it contains a behavior that stays forever in state s_1:



[2]if we had *all* linear behaviors of a system, we could in principle reconstruct its branching behavior as well

Branching-Time Temporal Logics

The “solutions” (models) of a formula are states of a transition system (Kripke structure).

CTL (Computation Tree Logic) – Syntax

CTL formulas are defined by the following grammar:

$$
\begin{array}{rcl}
\phi & ::= & p \mid q \mid \ldots \\
 & & \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \\
 & & \mid EG\phi_1 \mid AG\phi_1 \\
 & & \mid EX\phi_1 \mid AX\phi_1 \\
 & & \mid E(\phi_1 U \phi_2) \mid A(\phi_1 U \phi_2)
\end{array}
$$

E ("there exists a path") and A ("for all paths") are called *path quantifiers*. They are sometimes written $\exists$ and $\forall$.

CTL – Semantics: Intuition

Let s be a state of the Kripke structure.

$$s \models EG\phi$$

iff there exists a path starting from s such that the corresponding trace σ satisfies

$$\sigma \models G\phi$$

CTL – Semantics: Intuition

Let s be a state of the Kripke structure.

$$s \models EG\phi$$

iff there exists a path starting from s such that the corresponding trace σ satisfies

$$\sigma \models G\phi$$

Note:

- The 2nd $\models$ is the LTL satisfaction relation.
- The 1st $\models$ refers to the CTL satisfaction relation.
- To be more pedantic:

$$s \models_{CTL} EG\phi \qquad \text{iff} \qquad \sigma \models_{LTL} G\phi$$

CTL – Semantics: Intuition

Let s be a state of the Kripke structure.

$$s \models AG\phi$$

iff

CTL – Semantics: Intuition

Let s be a state of the Kripke structure.

$$s \models AG\phi$$

iff for every path starting from s, the corresponding trace σ satisfies

$$\sigma \models G\phi$$

CTL – Semantics: Intuition

Let s be a state of the Kripke structure.

$$s \models AG\phi$$

iff for every path starting from s, the corresponding trace σ satisfies

$$\sigma \models G\phi$$

Quiz: do we need $EF\phi$? can we express it in terms of the CTL modalities shown earlier?

CTL – Semantics

Let (P, S, S_0, L, R) be a Kripke structure and let $s \in S$.
A trace starting from s is a sequence $\sigma = \sigma_0, \sigma_1, \cdots$, such that there is a path $s = s_0, s_1, \cdots$ starting from s, and $\sigma_i = L(s_i)$ for all i.

Satisfaction relation for CTL:

$s \models p$	iff	$p \in L(s)$
$s \models \phi_1 \wedge \phi_2$	iff	$s \models \phi_1$ and $s \models \phi_2$
$s \models \neg\phi$	iff	$s \not\models \phi$
$s \models EG\phi$	iff	$\exists$trace σ starting from s : $\sigma \models_{LTL} G\phi$
$s \models AG\phi$	iff	$\forall$traces σ starting from s : $\sigma \models_{LTL} G\phi$
$s \models EX\phi$	iff	$\exists$trace σ starting from s : $\sigma \models_{LTL} X\phi$
$s \models E(\phi_1 U \phi_2)$	iff	$\exists$trace σ starting from s : $\sigma \models_{LTL} \phi_1 U \phi_2$

...

CTL – Examples

How to express these properties in CTL?

"p *holds at all reachable states*"

CTL – Examples

How to express these properties in CTL?

"p *holds at all reachable states*" AGp

CTL – Examples

How to express these properties in CTL?

"p holds at all reachable states" AGp

"there exists a way to get back to the initial state from any reachable state"

CTL – Examples

How to express these properties in CTL?

"p holds at all reachable states" AGp

"there exists a way to get back to the initial state from any reachable state" $AG\ EF$ *init*

Bibliography

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

Clarke, E., Grumberg, O., and Peled, D. (2000).
Model Checking.
MIT Press.

Emerson, E. A. (1990).
Handbook of theoretical computer science (vol. b).
chapter Temporal and modal logic, pages 995–1072. MIT Press.

Huth, M. and Ryan, M. (2004).
Logic in Computer Science: Modelling and Reasoning about Systems.
Cambridge University Press.

Manna, Z. and Pnueli, A. (1991).
The Temporal Logic of Reactive and Concurrent Systems: Specification.
Springer-Verlag, New York.

Manna, Z. and Pnueli, A. (1995).
Temporal Verification of Reactive Systems: Safety.
Springer-Verlag, New York.

Pnueli, A. (1981).
A temporal logic of concurrent programs.
Theoretical Computer Science, 13:45–60.