CS-E3220 Declarative Progamming

Modal Logics and Model-Checking

Jussi Rintanen

Department of Computer Science
Aalto University

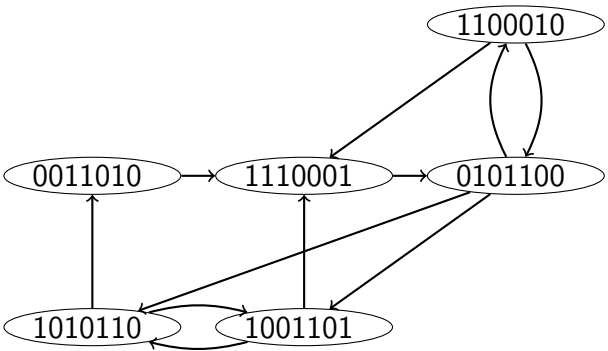
October 30, 2019

Transition System Model

Specification languages for Verification and Validation

- Modeling of Systems
  - Transition system models
  - Behavior of system, without reference to what is being done with it
  - Can be used for validation, control, diagnosis, monitoring, ...
- Modeling of Correctness properties
  - What properties the system's behavior should have?
  - Properties of reachable states
  - Properties of executions

Transition System Model



- States
- Transition relation

Transition System Model: Modeling Languages

- Languages based on transition rules:
  - Petri nets (place-transition nets)
  - languages used in AI planning research: STRIPS, ADL, PDDL, ...
  - ...
- Languages based on communicating automata/processes
  - UPPAAL
  - SMV
  - PROMELA
  - ...

  Transitions in individual automata/processes expressed as rules

Transition System Model: Applications

Tools

- NuSMV: model-checker (OBDD, SAT, ...)
- SPIN
- UPPAAL

Application areas

- computing HW (CPUs)
- communication protocols
- avionics SW (aircraft/spacecraft)
- distributed systems

Typical features:

- Focus on Boolean and numeric state variables
- HW, abstracted SW, Discrete & Hybrid systems (SW + physical)

Specification Languages

- Modeling languages for systems
  - How does the system behave, step by step
  - Focus on each possible action/event in separation of others
- Modeling languages for correctness properties
  - What properties should the executions of the system have
  - Statements about long sequences of events in the system
  - Properties:
    - Deadlocks: Can the system get in a state in which no event is possible
    - Livelocks: System stuck doing useless work without progress
    - Complex properties about the system's state in one time point
    - Complex properties about the system's state over time

Specification Languages

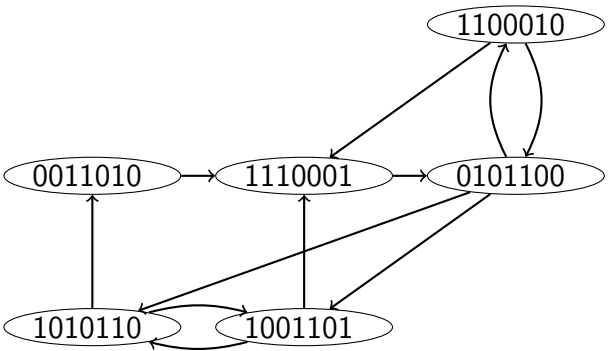
Next: fundamentals of specification languages with features for

- time
- networks, graphs

Modal Logics are the framework on which such languages are based.

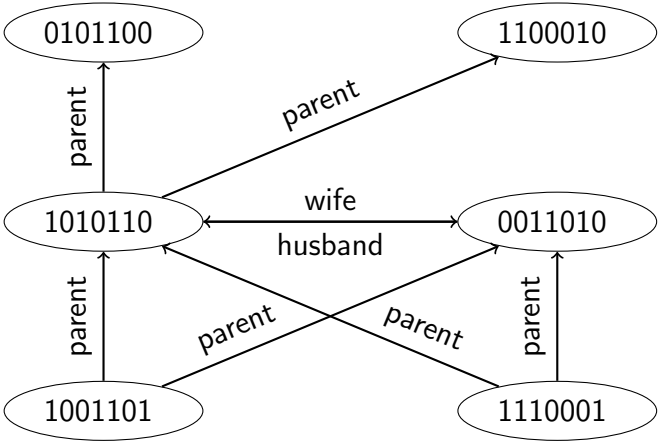
- application: specification languages with time
- application: specification languages with relational features
- application: semantic web (description logics, OWL, ...)

Transition System Model



- States
- Transition relation

Networked Model of Data



- Objects
- Relations on objects
- Properties of objects

Modal Logics

- Propositional logic $\sim$ Boolean functions of facts (in one "situation")
- Modal logics are about multiple "situations" or "worlds"
- Introduce modal operators $\Box$, $\Diamond$, ...
- These are quantification over alternative worlds: $\Box\phi$ can mean
  - ϕ is necessary $= \phi$ true in all possible worlds
  - ϕ is believed $= \phi$ true in all worlds considered possible
  - ϕ is known $= \phi$ true in all worlds considered possible (including the actual)
  - ϕ is obligatory $= \phi$ true in all worlds considered acceptable/legal
  - ϕ is always true $= \phi$ true in all future worlds

Modal Logics

- $K\phi$: It is known that ϕ (Epistemic Logic)
- $B\phi$: It is believed that ϕ (Epistemic Logic)
- $\Box\phi$: It is necessary that ϕ (Alethic logic)
- $\Box\phi$: It is obligatory that ϕ (Deontic logic)
- $\mathcal{G}\phi$: Always in the future ϕ (Temporal Logic)
- $[\pi]\phi$: After executing π necessarily ϕ (Dynamic Logic)

Modal Logics

In the propositional logic, the truth of $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$ etc. determined uniquely by the truth of ϕ_1 and ϕ_2.

Example

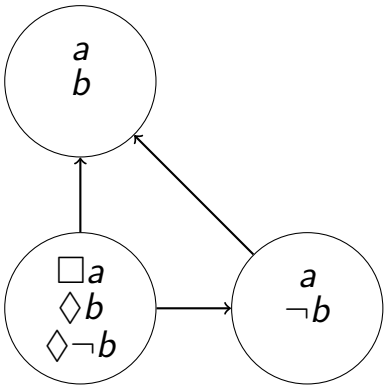
If p is true, can we say whether Bp is true or false?

What about operators K, $\Box$, $\mathcal{G}$ and $[\pi]$ from the previous slide?

Modal logics are not truth-functional!

Modal Logics: The Kripke Semantics

Modal logics have possible worlds semantics (Saul Kripke)



Formula $\Box\phi$ is true in a world if ϕ is true in all accessible worlds

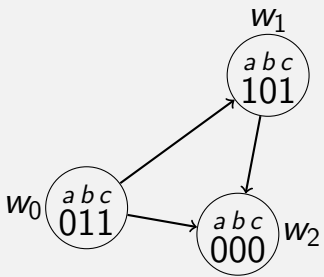
Formula $\Diamond\phi$ is true in a world if ϕ is true in at least one accessible worlds

Semantics of Modal Logics

Models in the Propositional Logic

$M = \overset{a\ b\ c}{011}$ $M \models \phi$ for any ϕ evaluated truth-functionally

Models in Modal Logics



- $\wedge$, $\vee$, $\neg$ evaluated truth-functionally
- $M \models_w \Box\phi$ iff ϕ true in all worlds accessible from w
- Examples:
  - $M \models_{w_0} \Box\neg b$
  - $M \models_{w_0} \neg\Box a$
  - $M \models_{w_0} \neg\Box\neg a$

Kripke Semantics Formally

Definition (Kriple Model (W, R, X, v))

- W is a set of worlds,
- $R \subseteq W \times W$ is the accessibility relation,
- X is the set of propositional variables,
- $v : W \times X \to \{0, 1\}$ is the valuation.

$$
\begin{aligned}
M \models_w x &\quad \text{iff } v(w, x) = 1 \\
M \models_w \neg\phi &\quad \text{iff } M \not\models_w \phi \\
M \models_w \phi_1 \vee \phi_2 &\quad \text{iff } M \models_w \phi_1 \text{ or } M \models_w \phi_2 \\
M \models_w \phi_1 \wedge \phi_2 &\quad \text{iff } M \models_w \phi_1 \text{ and } M \models_w \phi_2 \\
M \models_w \Box\phi &\quad \text{iff } M \models_{w'} \phi \text{ for all } w' \text{ such that } wRw' \\
M \models_w \Diamond\phi &\quad \text{iff } M \models_{w'} \phi \text{ for some } w' \text{ such that } wRw'
\end{aligned}
$$

Propositional Dynamic Logic

- Logic about regular paths in a graph
  - Originally: Regular expression $\sim$ program with conditionals, loops, ...
  - Nowadays: also other interpretations (e.g. in knowledge representation)
- Infinitely many operators $[\pi]$ and $\langle\pi\rangle$, for all regular expressions π
  - atomic programs a
  - sequential composition $\pi_1; \pi_2$
  - nondeterministic choice $\pi_1 \cup \pi_2$
  - iteration π_1*
  - test $\phi?$
- Examples:

$[a]\phi$	ϕ necessarily holds after atomic program a has been executed
$[a \cup b]\phi$	ϕ necessarily holds after either a or b has been executed
$\langle a; b\rangle\phi$	ϕ possibly holds after a and b have been executed

Propositional Dynamic Logic

Common programming language constructs expressible in PDL:

$$
\begin{aligned}
\text{if } \varphi \text{ then } \alpha \text{ else } \beta &\equiv (\varphi?;\alpha) \cup (\neg\varphi?;\beta) \\
\text{while } \varphi \text{ do } \alpha &\equiv (\varphi?;\alpha)*;\neg\varphi? \\
\text{repeat } \alpha \text{ until } \varphi &\equiv \alpha;(\neg\varphi?;\alpha)*;\varphi?
\end{aligned}
$$

Propositional Dynamic Logic vs. Description Logics

concept	description logic	dynamic logic
"is a grandparent"	$\exists child.\exists child.\top$	$\langle child; child\rangle\top$
"unmarried"	$\neg\exists spouse.\top$	$\neg\langle spouse\rangle\top$
"bachelor"	$male \sqcap \neg\exists spouse.\top$	$male \wedge \neg\langle spouse\rangle\top$
"has an ancestor with PhD"	-	$\langle parent; parent*\rangle$PhD

Correspondence between DL and PDL:

- Some operators match exactly:
  - $\forall R.C$ and $\exists R.C$ correspond to $[R]C$ and $\langle R\rangle C$
  - $\sqcup$ and $\sqcap$ correspond to $\vee$ and $\wedge$
- Some DL features have no match in PDL:
  - E.g. cardinality ("person with more than 3 children")
  - Not all DLs have arbitrary regular expressions as *roles*

PDL: Accessibility Relations of Programs

Given accessibility relations $R_a \subseteq W \times W$ for atomic programs $a \in A$, define accessibility relations for all programs.

Let α and β have accessibility relations R_α and R_β, respectively. Then

1. $R_{\alpha \cup \beta} = R_\alpha \cup R_\beta$
2. $R_{\alpha;\beta} = \pi_{1,3}(R_\alpha \bowtie R_\beta) = \{(w, w') | u \in W, wR_\alpha u, uR_\beta w'\}$
3. $R_{\alpha *} = (R_\alpha)*$
4. $R_{\varphi ?} = \{(w, w) | w \in W, M \models_w \varphi\}$

Note: Reflexive transitive closure $R*$ of R is the smallest R' such that $R \subseteq R'$, and $\{(w, w) | w \in W\} \subseteq R'$, and if $(w_1, w_2) \in R'$ and $(w_2, w_3) \in R'$, then $(w_1, w_3) \in R'$.

Propositional Dynamic Logic

Definition (Models in PDL)

A model in the propositional dynamic logic is a tuple $M = \langle W, R_{a_1}, \ldots, R_{a_n}, A, X, v \rangle$ where

- W is a set of possible worlds,
- A is a set of atomic programs,
- R_a for every $a \in A$ is the accessibility relation for a,
- X is the set of propositional variables, and
- $v : W \times X \to \{0, 1\}$ is a valuation that assigns truth values to propositional variables in every possible world.

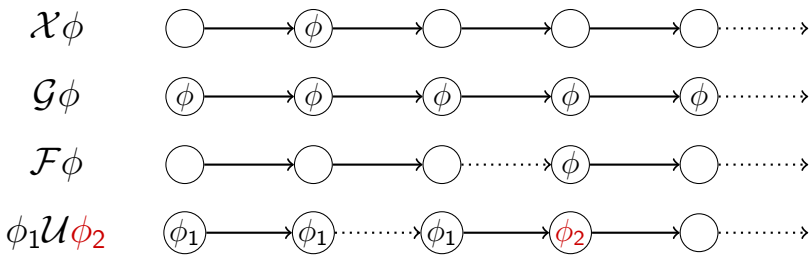
$M \models_w [\alpha]\varphi$ *iff* $M \models_{w'} \varphi$ *for every* $w' \in W$ *such that* $wR_\alpha w'$.

Temporal Logics

- Linear Time Temporal Logic
  - Statements about a single infinite sequence of states
- Branching time logics: Computation Tree Logics CTL and CTL$*$
  - Statements about an infinite branching tree of states
  - Path quantification: On all paths starting in state/node
  - Path quantification: On some path starting in state/node

Linear Temporal Logic

- $\mathcal{G}\phi$: ϕ holds in all future times (Always)
- $\mathcal{F}\phi$: ϕ holds in at least one future time (Eventually)
- $\phi_1 \mathcal{U} \phi_2$: ϕ_1 holds until ϕ_2 holds
- $\mathcal{X}\phi$: ϕ holds in the next time



LTL Examples

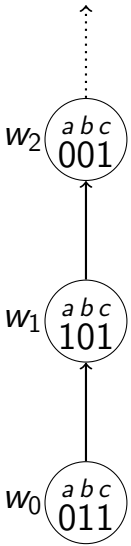
- Traffic lights are green infinitely often
- Only one light on in any direction
- After green there will eventually be red
- Red light is followed by Yellow (with no Green in between)
- Green lights for crossing traffic never simultaneous

$\mathcal{G}\mathcal{F}(green_1)$
$\mathcal{G}(\neg(green_1 \wedge red_1) \wedge \neg(green_1 \wedge yellow_1) \wedge \neg(yellow_1 \wedge red_1))$
$\mathcal{G}(green_1 \rightarrow F(red_1))$
$\mathcal{G}(red_1 \rightarrow (red_1 U yellow_1))$
$\mathcal{G}\neg(green_1 \wedge green_2)$

LTL Applications

- Model-Checking: Given a transition system and an LTL formula ϕ: Do all transition sequences satisfy ϕ?
- Program synthesis: Find a controller for transition system so that all executions satisfy ϕ
- Control information in decision-making: Limit search to transition sequences that satisfy ϕ

Semantics of Linear Temporal Logic



- Connectives $\wedge$, $\vee$, $\neg$ evaluated truth-functionally
- $M \models_w \mathcal{G}\phi$ iff ϕ true in all future worlds
- $M \models_w \mathcal{F}\phi$ iff ϕ true in some future world
- $M \models_w \psi \mathcal{U} \phi$ iff ϕ true in the future, and ψ true until then
- $M \models_w \mathcal{X}\phi$ iff ϕ is true at the next time
- Examples:
  - $M \models_{w_0} \mathcal{F}a$
  - $M \models_{w_0} \mathcal{F}\neg a$
  - $M \models_{w_0} \mathcal{G}c$
  - $M \models_{w_0} \mathcal{X}\mathcal{G}\neg b$

Semantics for Linear Temporal Logic

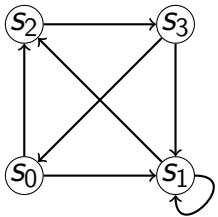
Definition

A linear temporal model $M = (X, \sigma)$ consists of propositional variables X and an infinite sequence of propositional valuations $\sigma = (v_0, v_1, v_2, \ldots)$

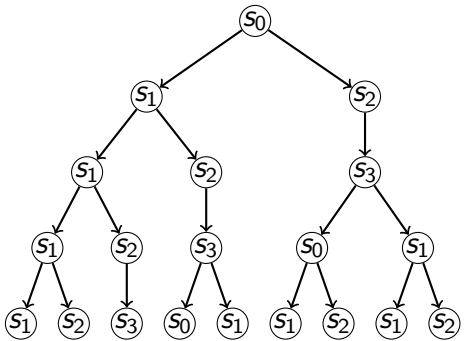
$M \models_i x$	iff $v_i(x) = 1$
$M \models_i \neg\phi$	iff $M \not\models_i \phi$
$M \models_i \phi_1 \vee \phi_2$	iff $M \models_i \phi_1$ or $M \models_i \phi_2$
$M \models_i \phi_1 \wedge \phi_2$	iff $M \models_i \phi_1$ and $M \models_i \phi_2$
$M \models_i \mathcal{G}\phi$	iff $M \models_j \phi$ for all $j \geq i$
$M \models_i \mathcal{F}\phi$	iff $M \models_j \phi$ for some $j \geq i$
$M \models_i \mathcal{X}\phi$	iff $M \models_{i+1} \phi$
$M \models_i \phi\mathcal{U}\psi$	iff for some $j \geq i, M \models_j \psi$ and $M \models_k \phi$ for all $k \in \{i, \ldots, j-1\}$

Computation Trees

Transition graph



Corresponding computation tree



Same subtree for every node repeats indefinitely...

Logics for Computation Trees

- Two options with LTL:
  1. truth of ϕ as "for some path in the tree, ϕ holds"
  2. truth of ϕ as "for all paths in the tree, ϕ holds"
- CTL$*$ = LTL + path quantification:
  - $A\phi$: ϕ holds on all paths that start from current node
  - $E\phi$: ϕ holds on some path that start from current node
  - example $A\mathcal{F}$*finalstate*: final state eventually reached on all computation paths
  - example $A\mathcal{G}\mathcal{F}$*progress*: progress made on all paths infinitely often
  - example $A\mathcal{G}E\mathcal{F}$*progress*: progress *possible* on all paths all the time
- CTL: E or A must immediately precede each temporal operator
  - example: $A(\mathcal{G}a \vee \mathcal{F}b)$ is a CTL$*$ formula but not a CTL formula
  - All formulas are state formulas. (No path formulas!)

Logics for Computation Trees

- Two options with LTL:
  1. truth of ϕ as "for some path in the tree, ϕ holds"
  2. truth of ϕ as "for all paths in the tree, ϕ holds"
- CTL$*$ = LTL + path quantification:
  - $A\phi$: ϕ holds on all paths that start from current node
  - $E\phi$: ϕ holds on some path that start from current node
  - example $A\mathcal{F}$*finalstate*: final state eventually reached on all computation paths
  - example $A\mathcal{G}\mathcal{F}$*progress*: progress made on all paths infinitely often
  - example $A\mathcal{G}E\mathcal{F}$*progress*: progress *possible* on all paths all the time
- CTL: E or A must immediately precede each temporal operator
  - example: $A(\mathcal{G}a \vee \mathcal{F}b)$ is a CTL$*$ formula but not a CTL formula
  - All formulas are state formulas. (No path formulas!)