4. Modal Logic & Verification

José Proença

System Verification (CC4084) 2024/2025

CISTER – U.Porto, Porto, Portugal

https://fm-dcc.github.io/sv2425

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

CISTER - Research Centre in Real-Time & Embedded Computing Systems

Syllabus

- Introduction to model-checking
- CCS: a simple language for concurrency
  - Syntax
  - Semantics
  - Equivalence
  - mCRL2: modelling
- Dynamic logic
  - Syntax
  - Semantics
  - Relation with equivalence
  - mCRL2: verification
- Timed Automata
  - Syntax
  - Semantics (composition, Zeno)
  - Equivalence
  - UPPAAL: modelling
- Temporal logics (LTL/CTL)
  - Syntax
  - Semantics
  - UPPAAL: verification
- Probabilistic and stochastic systems
  - Going probabilistic
  - UPPAAL: monte-carlo

Recall: What's in a logic?

A logic

A language

i.e. a collection of well-formed expressions to which meaning can be assigned.

A semantics

describing how language expressions are interpreted as statements about something.

A deductive system

i.e. a collection of rules to derive in a purely syntactic way facts and relationships among semantic objects described in the language.

Note

- a purely syntactic approach (up to the 1940's; the sacred form)
- a model theoretic approach (A. Tarski legacy)

Semantic reasoning: models

- sentences
- models & satisfaction: $\mathcal{M} \models \phi$
- validity: $\models \phi$ (ϕ is satisfied in every possible structure)
- logical consequence: $\Phi \models \phi$ (ϕ is satisfied in every model of Φ)
- theory: $Th\,\Phi$ (set of logical consequences of a set of sentences Φ)

Syntactic reasoning: deductive systems

Deductive systems $\vdash$

- sequents
- Hilbert systems
- natural deduction
- tableaux systems
- resolution
- $\cdots$

- derivation and proof
- deductive consequence: $\Phi \vdash \phi$
- theorem: $\vdash \phi$

Soundness & completeness

- A deductive system $\vdash$ is sound wrt a semantics $\models$ if for all sentences ϕ

$$\vdash \phi \implies \models \phi$$

(every theorem is valid)

- $\cdots$ complete ...

$$\models \phi \implies \vdash \phi$$

(every valid sentence is a theorem)

Consistency & refutability

For logics with negation and a conjunction operator

- A sentence ϕ is refutable if $\neg\phi$ is a theorem (i.e. $\vdash \neg\phi$)
- A set of sentences Φ is refutable if some finite conjunction of elements in Φ is refutable
- ϕ or Φ is consistent if it is not refutable.

Examples

$$\mathcal{M} \models \phi$$

- Propositional logic (logic of uninterpreted assertions; models are truth assignments)
- Equational logic (formalises equational reasoning; models are algebras)
- First-order logic (logic of predicates and quatification over structures; models are relational structures)
- **Modal logics**
- ...

Modal Logic

Modal logic (from P. Blackburn, 2007)

Over the years modal logic has been applied in many different ways. It has been used as a tool for reasoning about time, beliefs, computational systems, necessity and possibility, and much else besides.

These applications, though diverse, have something important in common: the key ideas they employ (flows of time, relations between epistemic alternatives, transitions between computational states, networks of possible worlds) can all be represented as simple graph-like structures.

Modal logics are

- tools to talk about relational, or graph-like structures.
- fragments of classical ones, with restricted forms of quantification ...
- ... which tend to be decidable and described in a pointfree notations.

Syntax

$$\phi \ ::= \ p \mid \mathsf{true} \mid \mathsf{false} \mid \neg\phi \mid \phi_1 \land \phi_2 \mid \phi_1 \to \phi_2 \mid \langle\alpha\rangle\,\phi \mid [\alpha]\,\phi$$

where $p \in \mathsf{PROP}$ and $\alpha \in \mathsf{ACT}$

Disjunction ($\vee$) and equivalence ($\leftrightarrow$) are defined by abbreviation.

The *signature* of the basic modal language is determined by sets:

- PROP of propositional symbols (typically assumed to be denumerably infinite) and
- ACT of structured actions (or programs), also called modality symbols.

Basic Modal Logic

Syntax

$$\phi \; ::= \; p \mid \text{true} \mid \text{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \langle\alpha\rangle\,\phi \mid [\alpha]\,\phi$$

where $p \in \mathsf{PROP}$ and $\alpha \in \mathsf{ACT}$

Ex. 4.1: Interpreting formulas

- $\langle \mathit{drinkCoffee} \rangle\, \mathit{energetic}$: I will now drink coffee and will be in an energetic state
- $[\mathit{drink}]\, \neg \mathit{thirsty}$: If I drink anything now, I will not be in a thirsty state

Syntax

$$\phi \ ::= \ p \mid \mathsf{true} \mid \mathsf{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \langle\alpha\rangle\,\phi \mid [\alpha]\,\phi$$

where $p \in \mathsf{PROP}$ and $\alpha \in \mathsf{ACT}$

Ex. 4.1: Interpreting formulas

- $\langle \mathit{drinkCoffee}\rangle\, \mathit{energetic}$: I will now drink coffee and will be in an energetic state
- $[\mathit{drink}]\, \neg \mathit{thirsty}$: If I drink anything now, I will not be in a thirsty state
- $[\mathit{something}^*]\, [\mathit{pressCoffee}]\, \langle \mathit{getCoffee}\rangle\, \mathsf{true}$:
  If do *something* any number of times, and then
  I *press the coffee* button, then
  I will *get my coffee* – and that's it.

Notes

- if there is only one modality in the signature (i.e., ACT is a singleton), write simply $\Diamond\phi$ and $\Box\phi$
- the language has some redundancy: in particular modal connectives are dual (as quantifiers are in first-order logic): $[\alpha]\,\phi$ is equivalent to $\neg\langle\alpha\rangle\,\neg\phi$

Example

Models as LTSs over Act.

$ACT = Act \qquad$ (sets of actions)

$\langle a\rangle\,\phi$ can be read as "*it must observe a, and ϕ must hold after that.*"

$[a]\,\phi$ can be read as "*if it observes a, then ϕ must hold after that.*"

$\mathcal{M}, s \models \phi$ – **what does it mean?**

Model definition

A model for the language is a pair $\mathcal{M} = \langle \mathcal{L}, V \rangle$, where

- $\mathcal{L} = \langle S, \text{ACT}, \longrightarrow \rangle$ is an LTS:
  - S is a non-empty set of states (or points)
  - ACT are the labels consisting of (structured) action symbols (or modality symbols)
  - $\longrightarrow \subseteq S \times \text{ACT} \times S$ is the transition relation
- $V : \text{PROP} \longrightarrow \mathcal{P}(S)$ is a valuation.

When ACT $= 1$

- $\Diamond\phi$ and $\Box\phi$ instead of $\langle\cdot\rangle\,\phi$ and $[\cdot]\,\phi$
- $\mathcal{L} = \langle S, \longrightarrow \rangle$ instead of $\mathcal{L} = \langle S, \text{ACT}, \longrightarrow \rangle$
- $\longrightarrow \subseteq S \times S$ instead of $\longrightarrow \subseteq S \times \text{ACT} \times S$

Safistaction: for a model $\mathcal{M}$ and a state s

$$
\begin{array}{lll}
\mathcal{M}, s \models \text{true} & & \\
\mathcal{M}, s \not\models \text{false} & & \\
\mathcal{M}, s \models p & \text{iff} & s \in V(p) \\
\mathcal{M}, s \models \neg\phi & \text{iff} & \mathcal{M}, s \not\models \phi \\
\mathcal{M}, s \models \phi_1 \wedge \phi_2 & \text{iff} & \mathcal{M}, s \models \phi_1 \text{ and } \mathcal{M}, s \models \phi_2 \\
\mathcal{M}, s \models \phi_1 \rightarrow \phi_2 & \text{iff} & \mathcal{M}, s \not\models \phi_1 \text{ or } \mathcal{M}, s \models \phi_2 \\
\mathcal{M}, s \models \langle\alpha\rangle\,\phi & \text{iff} & \text{there exists } v \in S \text{ st } s \xrightarrow{\alpha} v \text{ and } \mathcal{M}, v \models \phi \\
\mathcal{M}, s \models [\alpha]\,\phi & \text{iff} & \text{for all } v \in S \text{ st } s \xrightarrow{\alpha} v \text{ and } \mathcal{M}, v \models \phi
\end{array}
$$

Satisfaction

A formula ϕ is

- satisfiable in a model $\mathcal{M}$ if it is satisfied at some point of $\mathcal{M}$
- globally satisfied in $\mathcal{M}$ ($\mathcal{M} \models \phi$) if it is satisfied at all points in $\mathcal{M}$
- valid ($\models \phi$) if it is globally satisfied in all models
- a semantic consequence of a set of formulas Γ ($\Gamma \models \phi$) if for all models $\mathcal{M}$ and all points s, if $\mathcal{M}, s \models \Gamma$ then $\mathcal{M}, s \models \phi$

Process logic (Hennessy-Milner logic)

- PROP $= \emptyset$ (hence $V = \emptyset$)
- $S = \mathcal{P}$ is a set states in a labelled transition system, typically process terms
- structured actions are built by the grammar $K := a \in Act \mid K + K$
- the underlying LTS is given by $\mathcal{L} = \langle \mathcal{P}, Act, \{\langle p, a, p' \rangle \mid a \in Act\} \rangle$

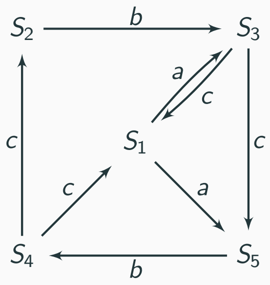
Satisfaction is abbreviated as

$$
\begin{array}{lll}
p \models \langle K \rangle \phi & \text{iff} & \exists_{q \in \{p' \mid p \xrightarrow{a} p' \,\wedge\, a \in K\}} \,.\, q \models \phi \\
p \models [K] \phi & \text{iff} & \forall_{q \in \{p' \mid p \xrightarrow{a} p' \,\wedge\, a \in K\}} \,.\, q \models \phi
\end{array}
$$

Process Logic Syntax (Hennessy-Milner Logic)

$$\phi \;::=\; \mathsf{true} \mid \mathsf{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \langle K\rangle\,\phi \mid [K]\,\phi$$

where $K := a \in Act \;\mid\; K + K$



Ex. 4.2: Prove:

1. $S_1 \models [a+b+c]\,(\langle b+c\rangle\,\mathsf{true})$
2. $S_2 \models [a]\,(\langle b\rangle\,\mathsf{true} \wedge \langle c\rangle\,\mathsf{true})$
3. $S_1 \not\models [a]\,(\langle b\rangle\,\mathsf{true} \wedge \langle c\rangle\,\mathsf{true})$
4. $S_2 \models [b]\,[c]\,(\langle a\rangle\,\mathsf{true} \vee \langle b\rangle\,\mathsf{true})$
5. $S_1 \models [b]\,[c]\,(\langle a\rangle\,\mathsf{true} \vee \langle b\rangle\,\mathsf{true})$
6. $S_1 \models [a+b]\,\langle b+c\rangle\,(\langle a\rangle\,\mathsf{true})$

$(P, <)$ a strict partial order with infimum 0

I.e., $P = \{0, a, b, c, \ldots\}$,
$a \rightarrow b$ means $a < b$,
$a < b$ and $b < c$ implies $a < c$
$0 < x$, for any $x \neq 0$
there are no loops
some elements may not be comparable

- $P, x \models \Box \text{false}$ if x is a maximal element of P
- $P, 0 \models \Diamond \Box \mathit{false}$ iff ...
- $P, 0 \models \Box \Diamond \Box \mathit{false}$ iff ...

Examples III

Temporal logic

- $\langle T, < \rangle$ where T is a set of time points (instants, execution states , ...) and $<$ is the earlier than relation on T.
- Thus, $\Box\varphi$ (respectively, $\Diamond\varphi$) means that φ holds in all (respectively, some) time points.

Epistemic logic (J. Hintikka, 1962)

- W is a set of agents
- $\alpha \models [K_i]\ \phi$ means that agent i always knows that ϕ is true.
- $\alpha \models \langle K_i \rangle\ \phi$ means that agent i can reach a state where he knows ϕ.
- $\alpha \models (\neg [K_i]\ \phi) \wedge (\neg [K_i]\ \neg\phi)$ means that agent i does not know whether ϕ is true or not.

Many variations exist, modelling knowledge and believes, knowledge of who knows what, distributed knowledge, etc.

Examples V

Deontic logic (G.H. von Wright, 1951)

- Obligations and permissions: must and can do.
- $\alpha \models \Box\ \phi$ means ϕ is obligatory.
- $\alpha \models \Diamond\ \phi$ means ϕ is a possibility.

Each logic accepts a different set of *principles* or *rules* (with variations), that makes their interpretation different.

Exercise

Ex. 4.3: Express the properties in Process Logic

- inevitability of a:
- progress (can always act):
- deadlock or termination (is stuck):

Ex. 4.4: What does this mean?

1. $\langle - \rangle\, \mathsf{false}$
2. $[-]\, \mathsf{true}$

"$-$" stands for $\sum_{a \in Act} a$, and "$-x$" abbreviates $\sum_{a \notin Act} a$

Recall syntax

$$
\begin{aligned}
\phi \;::=\;& \mathsf{true} \\
\mid\;& \mathsf{false} \\
\mid\;& \neg\phi \\
\mid\;& \phi_1 \wedge \phi_2 \\
\mid\;& \phi_1 \rightarrow \phi_2 \\
\mid\;& \langle K \rangle\, \phi \\
\mid\;& [K]\, \phi
\end{aligned}
$$

where $K := a \mid K + K$

Ex. 4.3: Express the properties in Process Logic

- inevitability of a: $\langle - \rangle\,\mathsf{true} \wedge [-a]\,\mathsf{false}$
- progress (can always act):
- deadlock or termination (is stuck):

Ex. 4.4: What does this mean?

1. $\langle - \rangle\,\mathsf{false}$
2. $[-]\,\mathsf{true}$

"$-$" stands for $\sum_{a \in Act} a$, and "$-x$" abbreviates $\sum_{a \notin Act} a$

Recall syntax

$$
\begin{aligned}
\phi \;::=\;& \mathsf{true} \\
\mid\;& \mathsf{false} \\
\mid\;& \neg\phi \\
\mid\;& \phi_1 \wedge \phi_2 \\
\mid\;& \phi_1 \rightarrow \phi_2 \\
\mid\;& \langle K \rangle\,\phi \\
\mid\;& [K]\,\phi
\end{aligned}
$$

where $K := a \mid K + K$

Express the following using Process Logic

Ex. 4.5: Coffee-machine

1. The user can have tea or coffee.
2. The user can have tea but not coffee.
3. The user can have tea after having 2 consecutive coffees.

Ex. 4.6: a's and b's

1. It is possible to do a after 3 b's, but not more than 1 a.
2. It must be possible to do a after [doing a and then b].
3. After doing a and then b, it is not possible to do a.

Ex. 4.7: Taxi network

- ϕ_0 = *In a taxi network, a car can collect a passenger or be allocated by the Central to a pending service*
- ϕ_1 = *This applies only to cars already on-service*
- ϕ_2 = *If a car is allocated to a service, it must first collect the passenger and then plan the route*
- ϕ_3 = *On detecting an emergency the taxi becomes inactive*
- ϕ_4 = *A car on-service is not inactive*

Process Logic with regular expressions

$$\phi \;::=\; \text{true} \mid \text{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \langle\alpha\rangle\,\phi \mid [\alpha]\,\phi$$

where $\alpha \in ACT$ are structured actions over a set Act:

$$\alpha := a \in Act \;\mid\; \alpha;\alpha \;\mid\; \alpha + \alpha \;\mid\; \alpha^*$$

More expressive than Process Logic. Used by mCRL2.

Process Logic with regular expressions

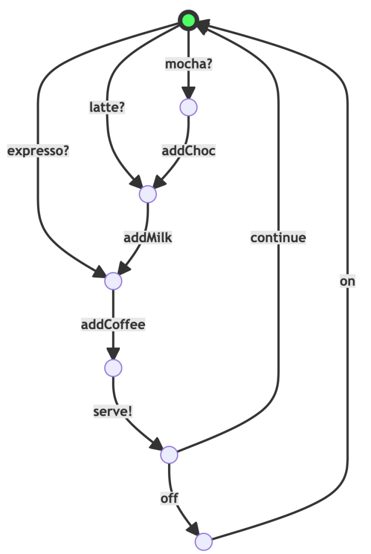
$$\phi \;::=\; \mathsf{true} \mid \mathsf{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \langle\alpha\rangle\,\phi \mid [\alpha]\,\phi$$

where $\alpha \in ACT$ are structured actions over a set Act:

$$\alpha := a \in Act \mid \alpha;\alpha \mid \alpha + \alpha \mid \alpha^*$$

Examples

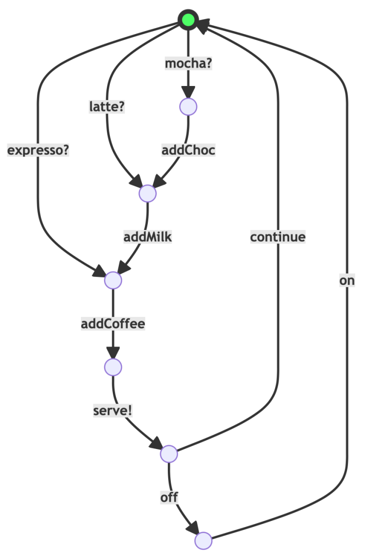
- "$\langle a.b.c\rangle$ true" means "$\langle a\rangle\langle b\rangle\langle c\rangle$ true"
- "$[a.b.c]$ false" means "$[a][b][c]$ false"
- "$\langle a^*.b\rangle$ true" means that b can be taken after some number of a's.
- "$\langle -^*.a\rangle$ true" means that a can eventually be taken
- "$[-^*]\langle a+b\rangle$ true" means it is always possible to do a or b



Ex. 4.8: What does this mean?

- $\langle -^*; serve! \rangle$ true
- $[-^*; (addChoc + addMilk); serve!]$ false
- $[-^*; addCoffee] \langle serve! \rangle$ true

- $\langle - \rangle$ true
- $[-^*] \langle - \rangle$ true
- $[-^*.a] \langle b \rangle$ true
- $[-^*.send] \langle (-send)^*.recv \rangle$ true



Ex. 4.9: Express using logic

1. The user can only have coffee after the coffee button is pressed.
2. The used must have coffee after the coffee button is pressed.
3. It is always possible to turn off the coffee machine.
4. It is always possible to reach a state where the coffee machine can be turned off.
5. It is never possible to add chocolate right after pressing the *latte button*.

mCRL2 Tools

Slides 3:

`https://fm-dcc.github.io/sv2425/slides/3-mcrl2.pdf`

Bisimulation and modal equivalence

Bisimulation (of models)

Definition

Given two models $\mathcal{M} = \langle \mathcal{L}, V \rangle$ and $\mathcal{M}' = \langle \mathcal{L}', V' \rangle$, a bisimulation of $\mathcal{L}$ and $\mathcal{L}'$ is also a bisimulation of $\mathcal{M}$ and $\mathcal{M}'$ if,

$$\text{whenever } s \; R \; s', \text{ then } V(s) = V'(s')$$

Invariance and definability

Lemma (invariance: bisimulation implies modal equivalence)

Given two models $\mathcal{M}$ and $\mathcal{M}'$, and a bisimulation R between their states:

if two states s, s' are related by R (i.e. sRs'),
then s, s' satisfy the same basic modal formulas.
(i.e., for all ϕ: $\mathcal{M}, s \models \phi \;\Leftrightarrow\; \mathcal{M}', s' \models \phi$)

Hence

Given 2 models $\mathcal{M}$ and $\mathcal{M}'$, if you can find ϕ such that

$\mathcal{M} \models \phi$ and $\mathcal{M}' \not\models \phi$

(or vice-versa) then they are NOT bisimilar.

Exercise

Ex. 4.10: Bisimilarity and modal equivalence

Consider the following transition systems:

$$\begin{tikzcd} 1 \arrow[loop above] \arrow[r] & 2 \end{tikzcd} \qquad\qquad \begin{tikzcd} 5 & \\ 3 \arrow[u] \arrow[d] \arrow[r, bend left] & 4 \arrow[l, bend left] \\ 6 & \end{tikzcd}$$

Give a modal formula that can be satisfied at point 1 but not at 3.

Ex. 4.11: Find distinguishing modal formula

1)

$$\begin{tikzcd} {} \arrow[r] & p_0 \arrow[r, bend left, "\text{coin}"] & p_1 \arrow[l, bend left, "\text{coffee}"] \end{tikzcd} \qquad \begin{tikzcd} {} \arrow[r] & q_0 \arrow[r, bend left, "\text{coin}"] & q_1 \arrow[l, "\text{tea}"'] \arrow[l, bend left, "\text{coffee}"] \end{tikzcd}$$

2)

$$\begin{tikzcd} & & p_1 \arrow[r, "b"] & p_2 \\ {} \arrow[r] & p_0 \arrow[ur, "a"] \arrow[dr, "a"'] & & \\ & & p_3 & \end{tikzcd} \qquad \begin{tikzcd} {} \arrow[r] & q_0 \arrow[r, "a"] & q_1 \arrow[r, "b"] & q_2 \end{tikzcd}$$

3)

$$\begin{tikzcd} & & p_1 \arrow[r, "b"] & p_2 \\ {} \arrow[r] & p_0 \arrow[ur, "a"] \arrow[dr, "a"'] & & \\ & & p_3 \arrow[r, "c"'] & p_4 \end{tikzcd} \qquad \begin{tikzcd} & & & q_2 \\ {} \arrow[r] & q_0 \arrow[r, "a"] & q_1 \arrow[ur, "b"] \arrow[dr, "c"'] & \\ & & & q_3 \end{tikzcd}$$

Richer modal logics

Richer modal logics

can be obtained in different ways, e.g.

- axiomatic extensions
- introducing more complex satisfaction relations
- support novel semantic capabilities
- ...

Examples

- richer temporal logics
- hybrid logic
- modal μ-calculus

Temporal Logics with $\mathcal{U}$ and $\mathcal{S}$

Until and Since

$$\mathcal{M}, s \models \phi\,\mathcal{U}\,\psi \quad \text{iff} \quad \text{there exists } r \text{ st } s \leq r \text{ and } \mathcal{M}, r \models \psi, \text{ and}$$
$$\text{for all } t \text{ st } s \leq t < r, \text{ one has } \mathcal{M}, t \models \phi$$

$$\mathcal{M}, s \models \phi\,\mathcal{S}\,\psi \quad \text{iff} \quad \text{there exists } r \text{ st } r \leq s \text{ and } \mathcal{M}, r \models \psi, \text{ and}$$
$$\text{for all } t \text{ st } r < t \leq s, \text{ one has } \mathcal{M}, t \models \phi$$

- Defined for temporal frames $\langle T, < \rangle$ (transitive, asymmetric).
- note the $\exists\forall$ qualification pattern: these operators are neither diamonds nor boxes.
- More general definition for other frames – it becomes more expressive than modal logics.

Temporal logics - rewrite using $\mathcal{U}$

- $\Diamond\psi =$
- $\Box\psi =$

Exercise

Temporal logics - rewrite using $\mathcal{U}$

- $\Diamond\psi = tt\,\mathcal{U}\,\psi$
- $\Box\psi =$

Exercise

Temporal logics - rewrite using $\mathcal{U}$

- $\Diamond\psi = tt\,\mathcal{U}\,\psi$
- $\Box\psi = \neg(\Diamond\neg\psi) = \neg(tt\,\mathcal{U}\,\neg\psi)$

Linear temporal logic (LTL)

$$\phi \;:=\; \text{true} \mid p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc\phi \mid \phi_1 \,\mathcal{U}\, \phi_2$$

mutual exclusion	$\square(\neg c_1 \vee \neg c_2)$
liveness	$\square\lozenge c_1 \wedge \square\lozenge c_2$
starvation freedom	$(\square\lozenge w_1 \rightarrow \square\lozenge c_1) \wedge (\square\lozenge w_1 \rightarrow \square\lozenge c_1)$
progress	$\square(w_1 \rightarrow \lozenge c_1)$
weak fairness	$\lozenge\square w_1 \rightarrow \square\lozenge c_1$
eventually forever	$\lozenge\square w_1$

- First temporal logic to reason about reactive systems [Pnueli, 1977]
- Formulas are interpreted over execution paths
- Express linear-time properties

state formulas to express properties of a state:

$$\Phi \;:=\; \text{true} \mid \Phi \wedge \Phi \mid \neg\Phi \mid \exists\psi \mid \forall\psi$$

path formulas to express properties of a path:

$$\psi \;:=\; \bigcirc\Phi \mid \Phi\,\mathcal{U}\,\Psi$$

mutual exclusion	$\forall \Box (\neg c_1 \vee \neg c_2)$
liveness	$\forall \Box \forall\Diamond c_1 \wedge \forall \Box \forall\Diamond c_2$
order	$\forall \Box (c_1 \vee \forall \bigcirc c_2)$

- Branching time structure encode transitive, irreflexive but not necessarily linear flows of time
- flows are trees: past linear; branching future

Motivation

Add the possibility of naming points and reason about their identity

Compare:

$$\Diamond(r \wedge p) \;\wedge\; \Diamond(r \wedge q) \;\;\rightarrow\;\; \Diamond(p \wedge q)$$

with

$$\Diamond(i \wedge p) \;\wedge\; \Diamond(i \wedge q) \;\;\rightarrow\;\; \Diamond(p \wedge q)$$

for $i \in \mathrm{NOM}$ (a nominal)

Syntax

$$\phi \;::=\; \ldots \mid p \mid \langle\alpha\rangle\,\phi \mid [\alpha]\,\phi \mid i \mid @_i\,\phi$$

where $p \in \mathrm{PROP}$ and $\alpha \in \mathrm{ACT}$ and $i \in \mathrm{NOM}$

Nominals i

- Are special propositional symbols that hold exactly on one state (the state they name)
- In a model the valuation V is extended from

$$V : \text{PROP} \longrightarrow \mathcal{P}(S)$$

to

$$V : \text{PROP} \longrightarrow \mathcal{P}(S) \quad \text{and} \quad V : \text{NOM} \longrightarrow S$$

where NOM is the set of nominals in the model

- Satisfaction:

$$\mathcal{M}, s \models i \qquad \text{iff } s = V(i)$$

The $@_i$ operator

$$
\begin{array}{lll}
\mathcal{M}, s \models \text{true} & & \\
\mathcal{M}, s \not\models \text{false} & & \\
\mathcal{M}, s \models p & \text{iff} & s \in V(p) \\
\mathcal{M}, s \models \neg\phi & \text{iff} & \mathcal{M}, s \not\models \phi \\
\mathcal{M}, s \models \phi_1 \wedge \phi_2 & \text{iff} & \mathcal{M}, s \models \phi_1 \text{ and } \mathcal{M}, s \models \phi_2 \\
\mathcal{M}, s \models \phi_1 \rightarrow \phi_2 & \text{iff} & \mathcal{M}, s \not\models \phi_1 \text{ or } \mathcal{M}, s \models \phi_2 \\
\mathcal{M}, s \models \langle\alpha\rangle\, \phi & \text{iff} & \text{there exists } v \in S \text{ st } s \xrightarrow{\alpha} v \text{ and } \mathcal{M}, v \models \phi \\
\mathcal{M}, s \models [\alpha]\, \phi & \text{iff} & \text{for all } v \in S \text{ st } s \xrightarrow{\alpha} v \text{ and } \mathcal{M}, v \models \phi \\
 & & \\
\mathcal{M}, s \models @_i\phi & \text{iff} & \mathcal{M}, u \models \phi \text{ and } u = V(i) \\
 & & [u \text{ is the state denoted by } i]
\end{array}
$$

Summing up

- basic hybrid logic is a simple notation for capturing the bisimulation-invariant fragment of first-order logic with constants and equality, i.e., a mechanism for equality reasoning in propositional modal logic.
- comes cheap: up to a polynomial, the complexity of the resulting decision problem is no worse than for the basic modal language