

## 4. Modal Logic & Verification

---

José Proença

System Verification (CC4084) 2025/2026

CISTER – U.Porto, Porto, Portugal

<https://fm-dcc.github.io/sv2526>



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

- 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 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)

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

## Deductive systems $\vdash$

- sequents
  - Hilbert systems
  - natural deduction
  - tableaux systems
  - resolution
  - ...
- 
- derivation and proof
  - deductive consequence:  $\Phi \vdash \phi$
  - theorem:  $\vdash \phi$

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

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

(every theorem is valid)

- ... **complete** ...

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

(every valid sentence is a theorem)

For logics with **negation** and a **conjunction** operator

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



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

- 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 **quantification** over structures; models are **relational structures**)
- **Modal logics**
- ...

# Modal Logic

---

*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

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

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

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.

## 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  $p \in \text{PROP}$  and  $\alpha \in \text{ACT}$

### Ex. 4.1: Interpreting formulas

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

## 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  $p \in \text{PROP}$  and  $\alpha \in \text{ACT}$

## Ex. 4.1: Interpreting formulas

- $\langle \text{drinkCoffee} \rangle \text{energetic}$ : I will now **drink coffee** and will be in an **energetic** state
- $[\text{drink}] \neg \text{thirsty}$ : If I **drink** anything now, I will not be in a **thirsty** state
- $[\text{something}^*] [\text{pressCoffee}] \langle \text{getCoffee} \rangle \text{true}$ :  
 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.

$ACT = Act$  (sets of actions)

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

$[a]\phi$  can be read as “**if** it observes  $a$ , then  $\phi$  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)
  - $\text{ACT}$  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 $\text{ACT} = 1$

- $\Diamond\phi$  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  $\longrightarrow \subseteq S \times \text{ACT} \times S$



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

$\mathcal{M}, s \models \text{true}$

$\mathcal{M}, s \not\models \text{false}$

$\mathcal{M}, s \models p$

iff  $s \in V(p)$

$\mathcal{M}, s \models \neg\phi$

iff  $\mathcal{M}, s \not\models \phi$

$\mathcal{M}, s \models \phi_1 \wedge \phi_2$

iff  $\mathcal{M}, s \models \phi_1$  and  $\mathcal{M}, s \models \phi_2$

$\mathcal{M}, s \models \phi_1 \rightarrow \phi_2$

iff  $\mathcal{M}, s \not\models \phi_1$  or  $\mathcal{M}, s \models \phi_2$

$\mathcal{M}, s \models \langle \alpha \rangle \phi$

iff there exists  $v \in S$  st  $s \xrightarrow{\alpha} v$  and  $\mathcal{M}, v \models \phi$

$\mathcal{M}, s \models [\alpha] \phi$

iff for all  $v \in S$  st  $s \xrightarrow{\alpha} v$  and  $\mathcal{M}, v \models \phi$

## Satisfaction

A formula  $\phi$  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  $\Gamma$  ( $\Gamma \models \phi$ ) 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)

- $\text{PROP} = \emptyset$  (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  $K := a \in \text{Act} \mid K + K$
- the underlying LTS is given by  $\mathcal{L} = \langle \mathcal{P}, \text{Act}, \{ \langle p, a, p' \rangle \mid a \in \text{Act} \} \rangle$

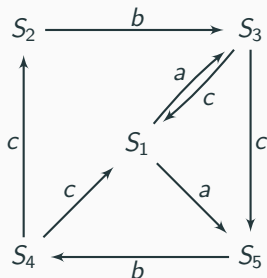
Satisfaction is abbreviated as

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

## Process Logic Syntax (Hennessy-Milner Logic)

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

where  $K := a \in \text{Act} \mid K + K$



### Ex. 4.2: Prove:

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

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

i.e.,  $P = \{0, a, b, c, \dots\}$ ,

$a \rightarrow b$  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  $P$
- $P, 0 \models \Diamond \Box \text{false}$  iff ...
- $P, 0 \models \Box \Diamond \Box \text{false}$  iff ...

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

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

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

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

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



## Ex. 4.3: Express the properties in Process Logic

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

## Ex. 4.4: What does this mean?

1.  $\langle - \rangle$  false
2.  $[-]$  true

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

## Recall syntax

$$\begin{aligned} \phi &::= \text{true} \\ &| \text{false} \\ &| \neg \phi \\ &| \phi_1 \wedge \phi_2 \\ &| \phi_1 \rightarrow \phi_2 \\ &| \langle K \rangle \phi \\ &| [K] \phi \end{aligned}$$

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

## Ex. 4.3: Express the properties in Process Logic

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

## Ex. 4.4: What does this mean?

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

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

## Recall syntax

$$\begin{aligned}
 \phi &::= \text{true} \\
 &| \text{false} \\
 &| \neg \phi \\
 &| \phi_1 \wedge \phi_2 \\
 &| \phi_1 \rightarrow \phi_2 \\
 &| \langle K \rangle \phi \\
 &| [K] \phi
 \end{aligned}$$

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

## 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

- $\phi_0 =$  In a taxi network, a car can *collect* a passenger or be *allocated* by the Central to a pending service
- $\phi_1 =$  This applies only to cars already *on-service*
- $\phi_2 =$  If a car is *allocated* to a service, it must first *collect* the passenger and then *plan* the route
- $\phi_3 =$  On detecting an *emergency* the taxi becomes inactive
- $\phi_4 =$  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

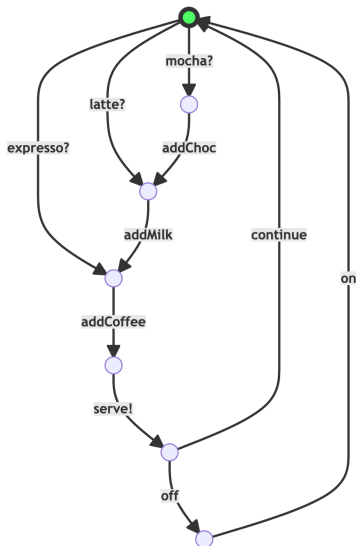
$$\phi ::= \text{true} \mid \text{false} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \langle \alpha \rangle \phi \mid [\alpha] \phi$$

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

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

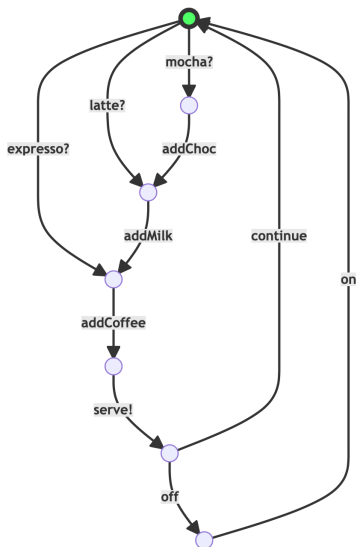
## Examples

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



## Ex. 4.8: What does this mean?

- $\langle -^*; \text{serve!} \rangle$  true
- $[-^*; (\text{addChoc} + \text{addMilk}); \text{serve!}]$  false
- $[-^*; \text{addCoffee}] \langle \text{serve!} \rangle$  true
- $\langle - \rangle$  true
- $[-^*] \langle - \rangle$  true
- $[-^*; a] \langle b \rangle$  true
- $[-^*; \text{send}] \langle (-\text{send})^*; \text{recv} \rangle$  true



### Ex. 4.9: Express using logic

1. The user **can only have coffee** after the **coffee button** is pressed.
2. The user **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

---

## 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,

whenever  $s R s'$ , then  $V(s) = V'(s')$

## 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  $\phi$ :  $\mathcal{M}, s \models \phi \Leftrightarrow \mathcal{M}', s' \models \phi$ )

## Hence

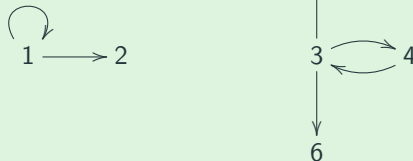
Given 2 models  $\mathcal{M}$  and  $\mathcal{M}'$ , if you can find  $\phi$  such that

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

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

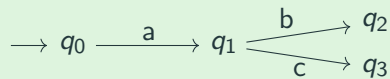
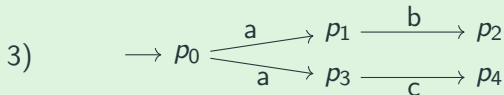
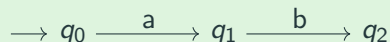
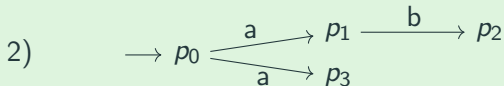
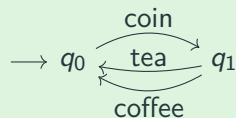
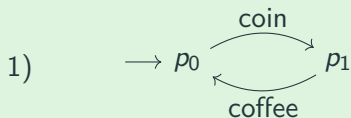
**Ex. 4.10: Bisimilarity and modal equivalence**

Consider the following transition systems:



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

## Ex. 4.11: Find distinguishing modal formula



To prove the converse of the invariance lemma requires passing to an **infinitary** modal language with arbitrary (countable) conjunctions and disjunctions. Alternatively, and more usefully, it can be shown for **finite** models:

## Lemma (modal equivalence implies bisimulation)

**If** two states  $s, s'$  from two finite models  $\mathcal{M} = \langle \langle S, R \rangle, V \rangle$  and  $\mathcal{M}' = \langle \langle S', R' \rangle, V' \rangle$  satisfy the same modal formulas,  
**then** there is a bisimulation  $B \subseteq S \times S'$  such that  $sBs'$ .

## Note

- The result can be weakened to image-finite models.
- Combining this result with the invariance lemma one gets the so-called modal equivalence theorem stating that, for image-finite models, bisimilarity and modal equivalence coincide. The result is also known as the Hennessy-Milner theorem who first proved it for process logics.

## Exercise

- Give an example of modally equivalent states in different Kripke structures which fail to be bisimilar.



## 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  $\mu$ -calculus

## Until and Since

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

$\mathcal{M}, s \models \phi \mathcal{S} \psi$       iff there **exists**  $r$  st  $r \leq s$  and  $\mathcal{M}, r \models \psi$ , and  
for **all**  $t$  st  $r < t \leq s$ , 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 $\mathcal{U}$

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

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

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

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

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

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

mutual exclusion	$\Box(\neg c_1 \vee \neg c_2)$
liveness	$\Box\Diamond c_1 \wedge \Box\Diamond c_2$
starvation freedom	$(\Box\Diamond w_1 \rightarrow \Box\Diamond c_1) \wedge (\Box\Diamond w_2 \rightarrow \Box\Diamond c_2)$
progress	$\Box(w_1 \rightarrow \Diamond c_1)$
weak fairness	$\Diamond\Box w_1 \rightarrow \Box\Diamond c_1$
eventually forever	$\Diamond\Box w_1$

- 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	$\forall \square (\neg c_1 \vee \neg c_2)$
liveness	$\forall \square \forall \Diamond c_1 \wedge \forall \square \forall \Diamond c_2$
order	$\forall \square (c_1 \vee \forall \bigcirc c_2)$

- 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  $i \in \text{NOM}$  (a **nominal**)

## Syntax

$$\phi ::= \dots \mid p \mid \langle \alpha \rangle \phi \mid [\alpha] \phi \mid i \mid @_i \phi$$

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

**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:

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

The @<sub>i</sub> operator
$$\mathcal{M}, s \models \text{true}$$

$$\mathcal{M}, s \not\models \text{false}$$

$$\mathcal{M}, s \models p$$

$$\text{iff } s \in V(p)$$

$$\mathcal{M}, s \models \neg \phi$$

$$\text{iff } \mathcal{M}, s \not\models \phi$$

$$\mathcal{M}, s \models \phi_1 \wedge \phi_2$$

$$\text{iff } \mathcal{M}, s \models \phi_1 \text{ and } \mathcal{M}, s \models \phi_2$$

$$\mathcal{M}, s \models \phi_1 \rightarrow \phi_2$$

$$\text{iff } \mathcal{M}, s \not\models \phi_1 \text{ or } \mathcal{M}, s \models \phi_2$$

$$\mathcal{M}, s \models \langle \alpha \rangle \phi$$

$$\text{iff there exists } v \in S \text{ st } s \xrightarrow{\alpha} v \text{ and } \mathcal{M}, v \models \phi$$

$$\mathcal{M}, s \models [\alpha] \phi$$

$$\text{iff for all } v \in S \text{ st } s \xrightarrow{\alpha} v \text{ and } \mathcal{M}, v \models \phi$$

$$\mathcal{M}, s \models @_i \phi$$

$$\text{iff } \mathcal{M}, u \models \phi \text{ and } u = V(i)$$

$$[u \text{ is the state denoted by } i]$$

## Increased frame definability

- **irreflexivity**:  $i \rightarrow \neg \Diamond i$
- **asymmetry**:  $i \rightarrow \neg \Diamond \Diamond i$
- **antisymmetry**:  $i \rightarrow \Box (\Diamond i \rightarrow i)$
- **trichotomy**:  $@_j \Diamond i \vee @_i j \vee @_i \Diamond j$

## 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