Lecture 4. Formal specifications: LTL, CTL

ELEC-E8110 Automation Systems Synthesis and Analysis

Igor Buzhinsky

igor.buzhinskii@aalto.fi

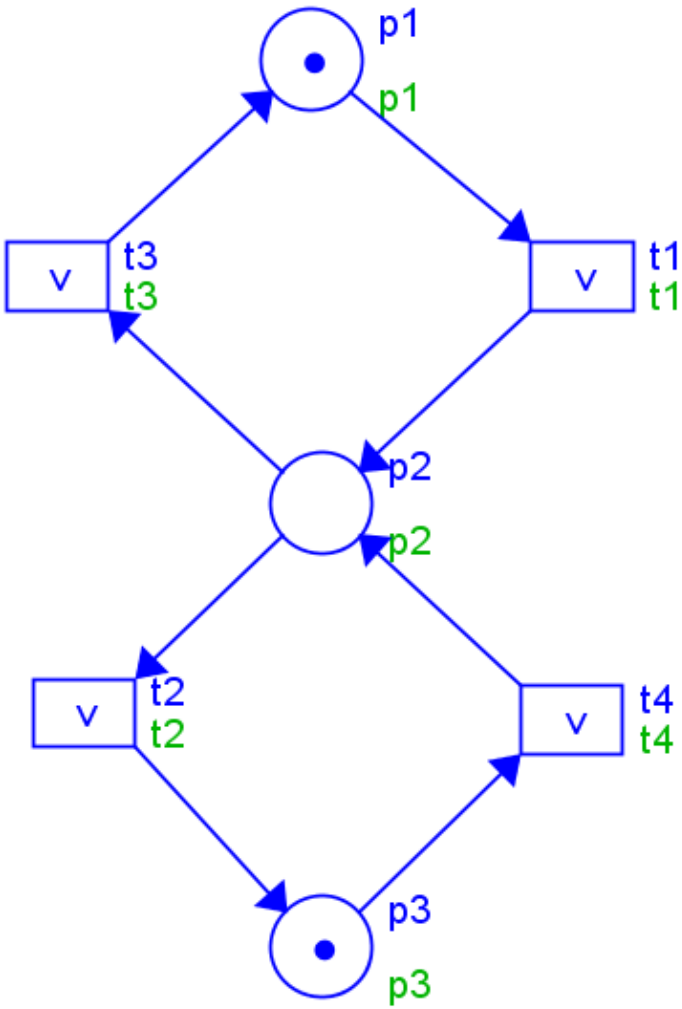
Aalto University
School of Electrical
Engineering

2018

State (reachability) graph of a system

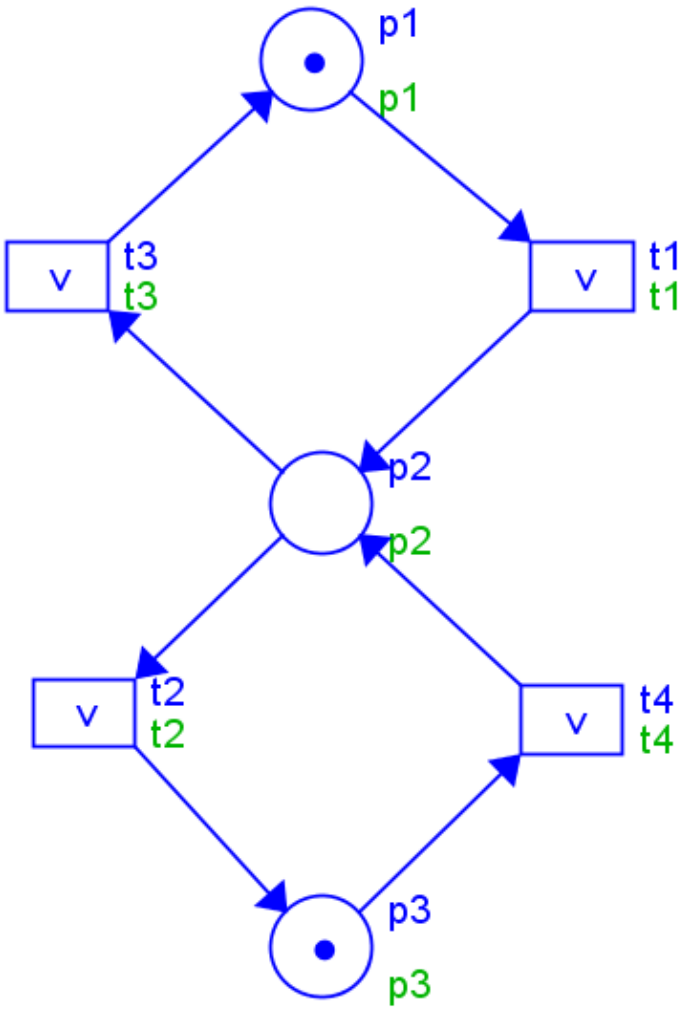
- Nodes: all reachable states of the system
- If the system is modular, then the state of the system consists of the state of all its modules
- Directed edges: one-step evolutions of the state
- Multiple outgoing edges are possible from each state, i.e. nondeterminism is common

State graph: example

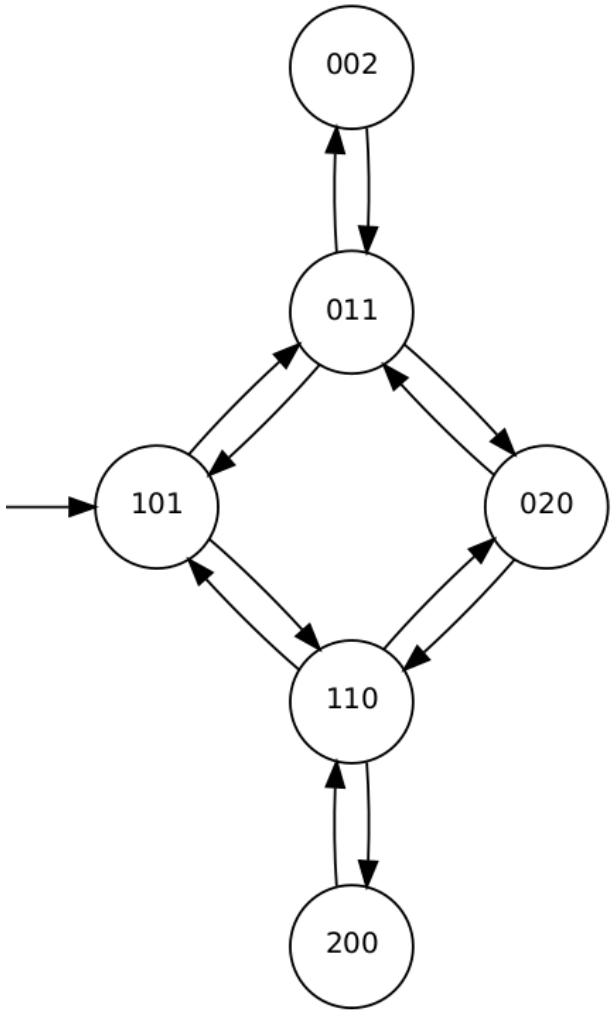


NCES module (actually, a Petri net)

State graph: example



NCES module (actually, a Petri net)

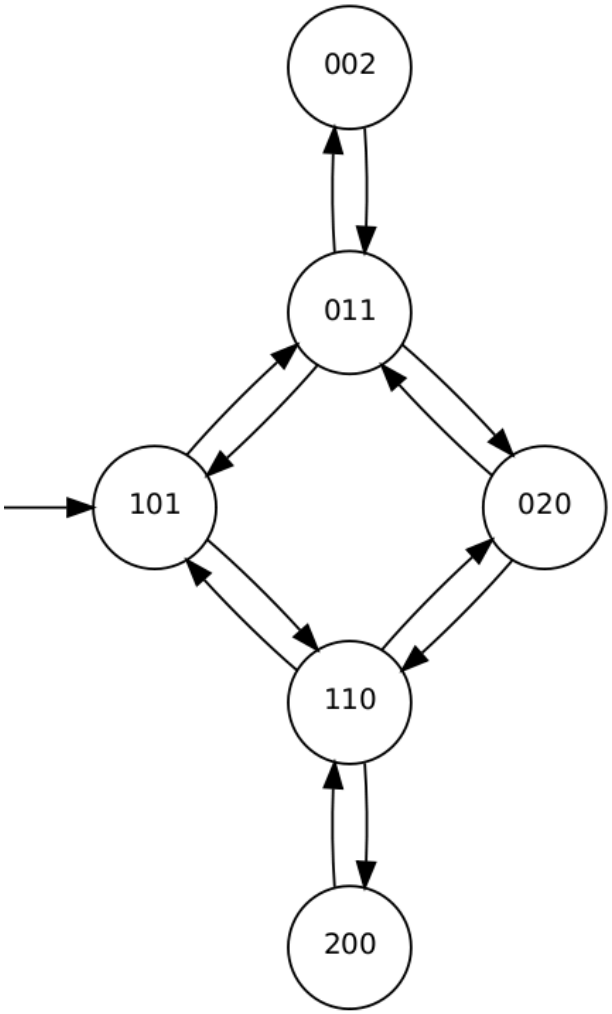


State graph, state $= p_1 p_2 p_3$

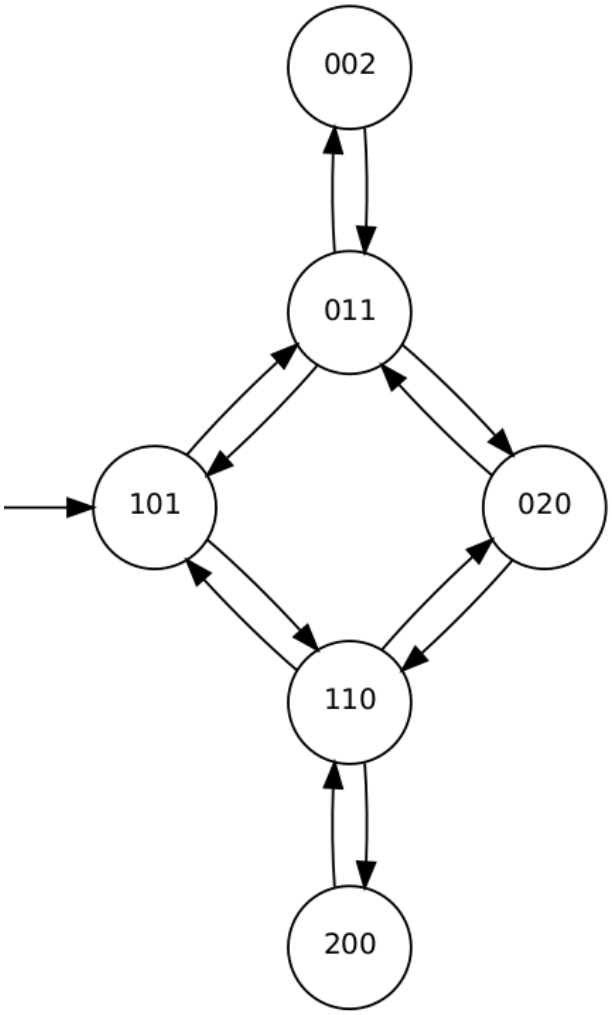
Kripke structures

- Formalization of a state graph
- Let AP be the a finite set of so-called **atomic propositions**
- Then $M = (S, I, T, L)$ is a **Kripke structure**, where:
- S is a finite set of **states**
- $I \subset S$ is a set of **initial states**
- $T \subset S \times S$ is a **transition relation**
- $L : S \to 2^{\mathrm{AP}}$ is a **labeling function**
- **No deadlock assumption:** $\forall s \in S \, \exists s' \in S : (s, s') \in T$

State graph interpreted as a Kripke structure

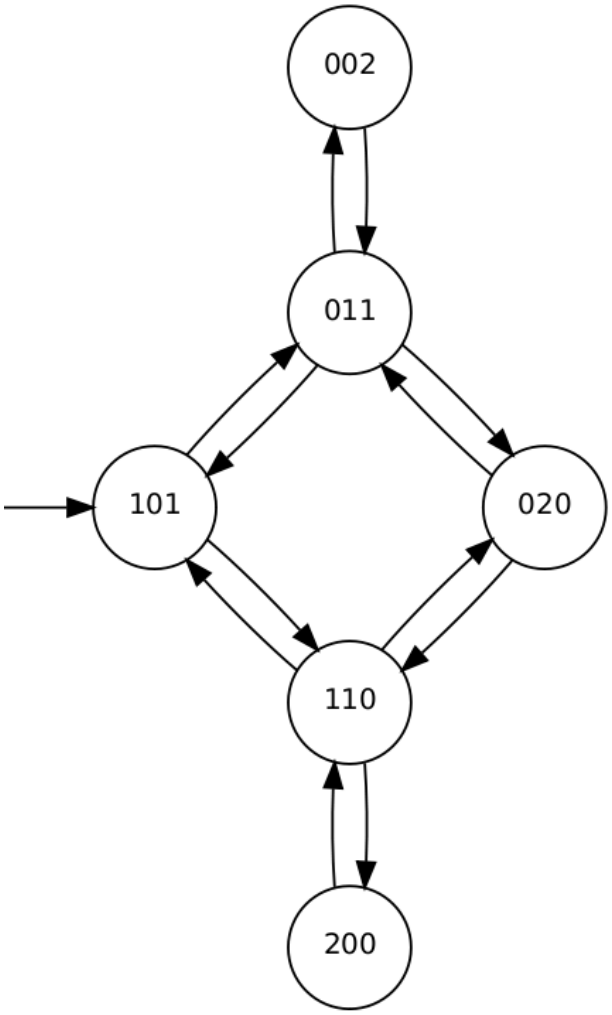


State graph interpreted as a Kripke structure



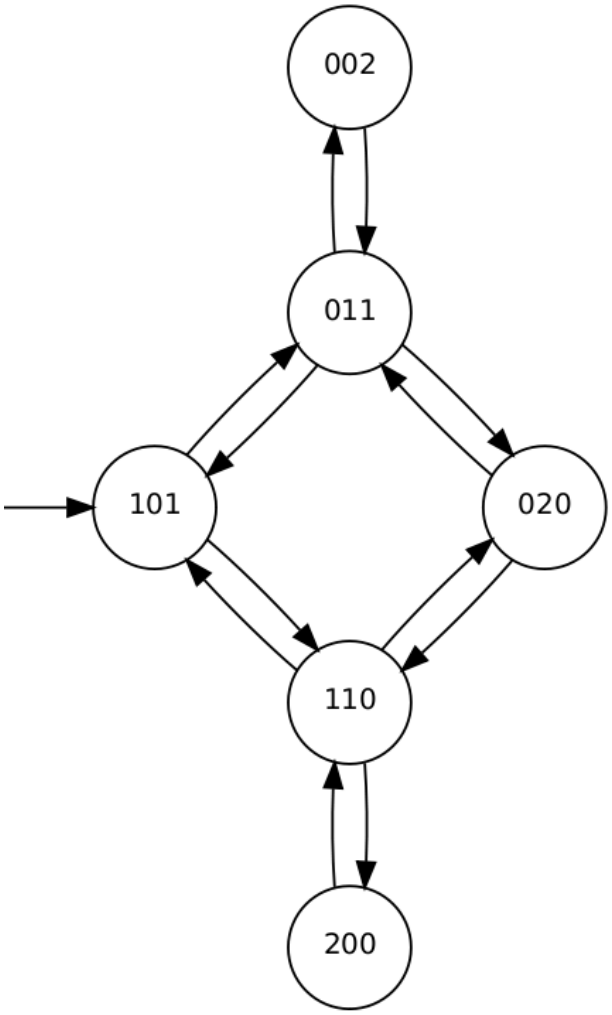
- AP =

State graph interpreted as a Kripke structure



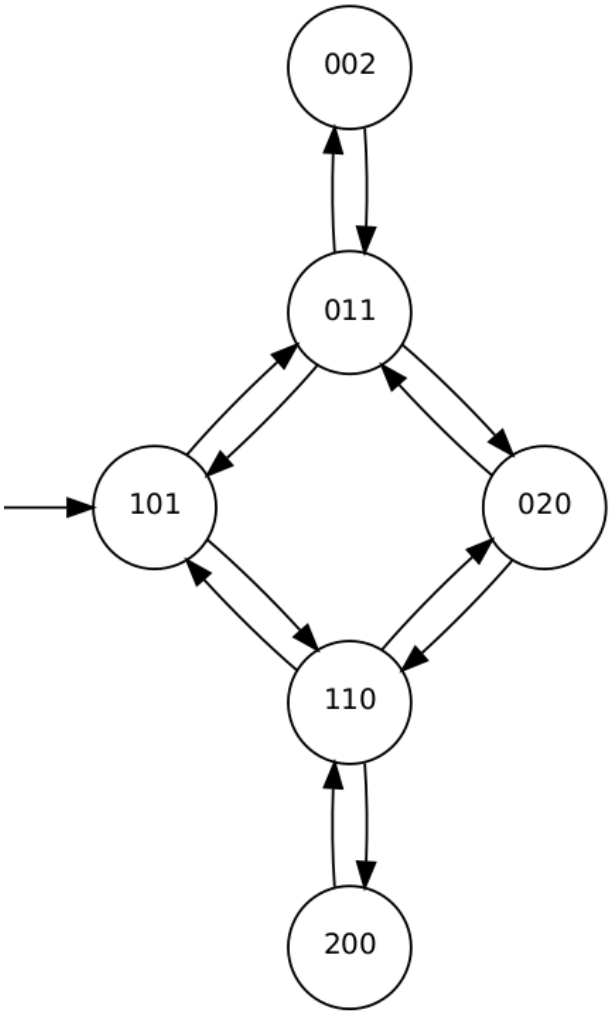
- AP $= \{\text{“}p_i = j\text{”} \mid i = 1..3, j = 0..2\}$
- S:

State graph interpreted as a Kripke structure



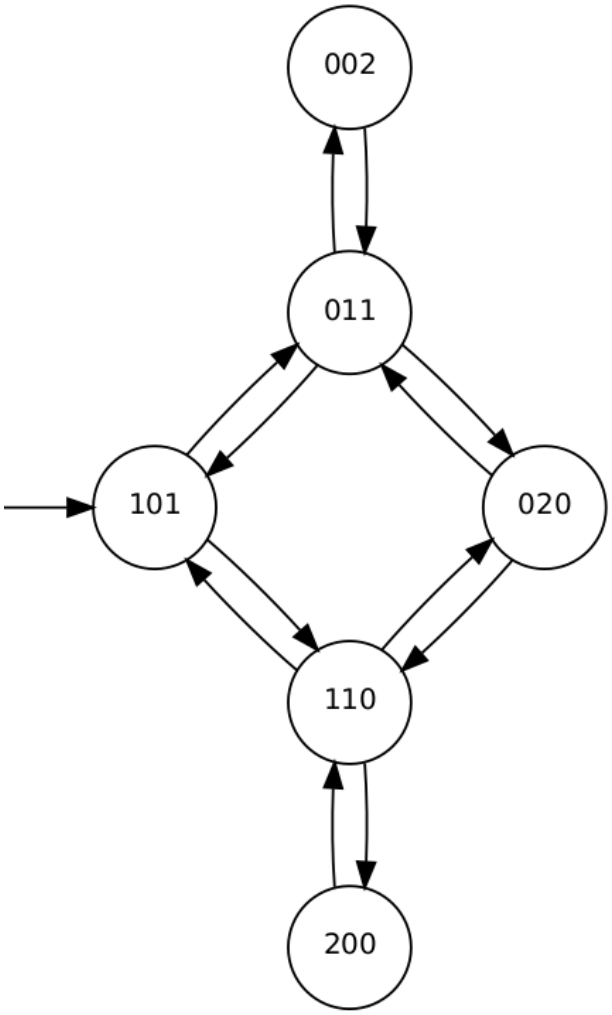
- AP $= \{\text{“}p_i = j\text{”} \mid i = 1..3, j = 0..2\}$
- S: nodes of this graph
- $I \subset S =$

State graph interpreted as a Kripke structure



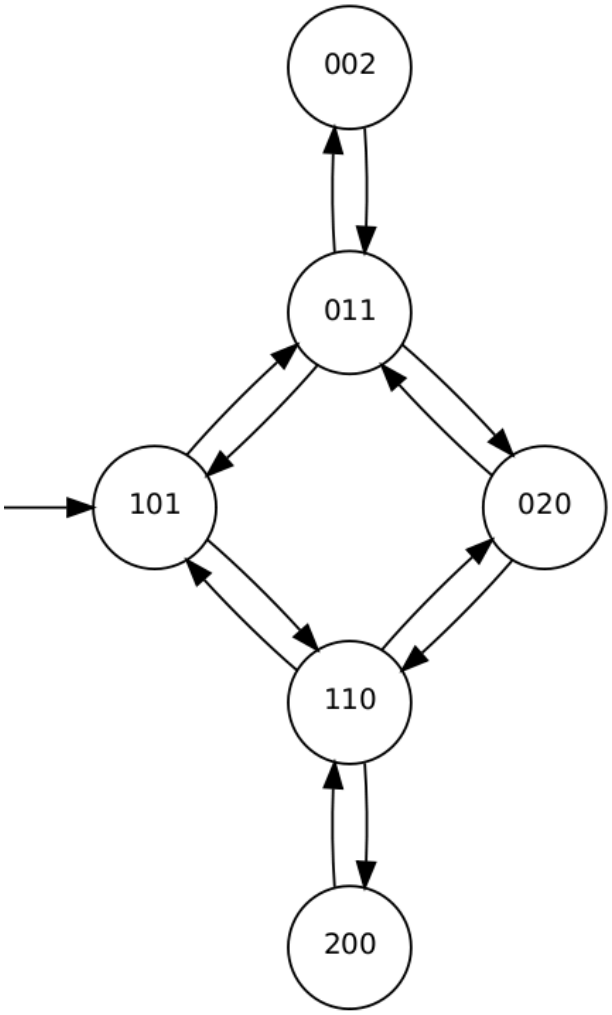
- AP $= \{\text{“}p_i = j\text{”} \mid i = 1..3, j = 0..2\}$
- S: nodes of this graph
- $I \subset S = \{101\}$
- Note: in UPPAAL and NCES models there is always one initial state!
- $T \subset S \times S$:

State graph interpreted as a Kripke structure



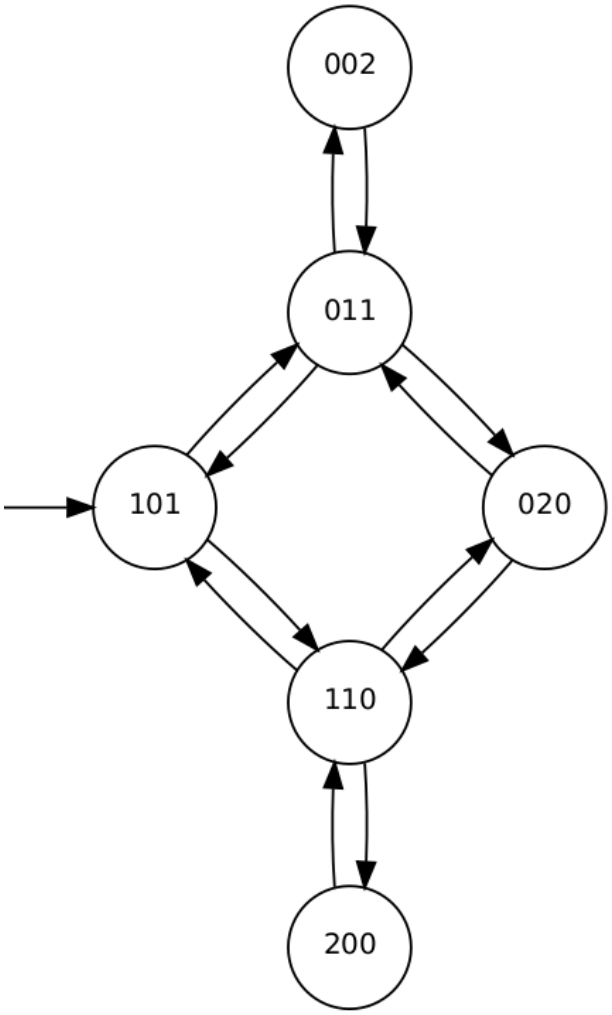
- $\mathrm{AP} = \{\text{“}p_i = j\text{”} \mid i = 1..3, j = 0..2\}$
- S: nodes of this graph
- $I \subset S = \{101\}$
- Note: in UPPAAL and NCES models there is always one initial state!
- $T \subset S \times S$: edges of the graph, e.g. $(002, 011)$, $(011, 002)$, ...
- $L : S \to 2^{\mathrm{AP}}$:

State graph interpreted as a Kripke structure



- $\mathrm{AP} = \{\text{“}p_i = j\text{”} \mid i = 1..3, j = 0..2\}$
- S: nodes of this graph
- $I \subset S = \{101\}$
- Note: in UPPAAL and NCES models there is always one initial state!
- $T \subset S \times S$: edges of the graph, e.g. $(002, 011)$, $(011, 002)$, ...
- $L : S \to 2^{\mathrm{AP}}$: token assignments (markings) in each state, e.g. $L(020) = \{\text{“}p_0 = 0\text{”}, \text{“}p_1 = 2\text{”}, \text{“}p_3 = 0\text{”}\}$

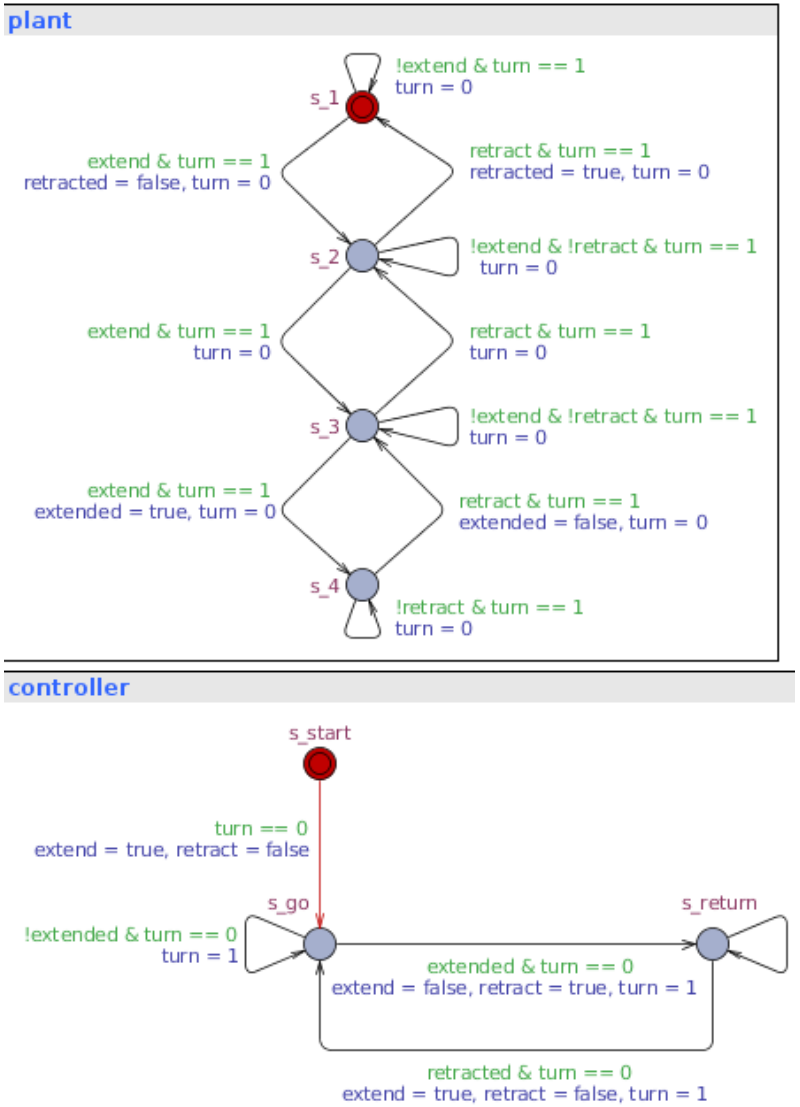
State graph interpreted as a Kripke structure



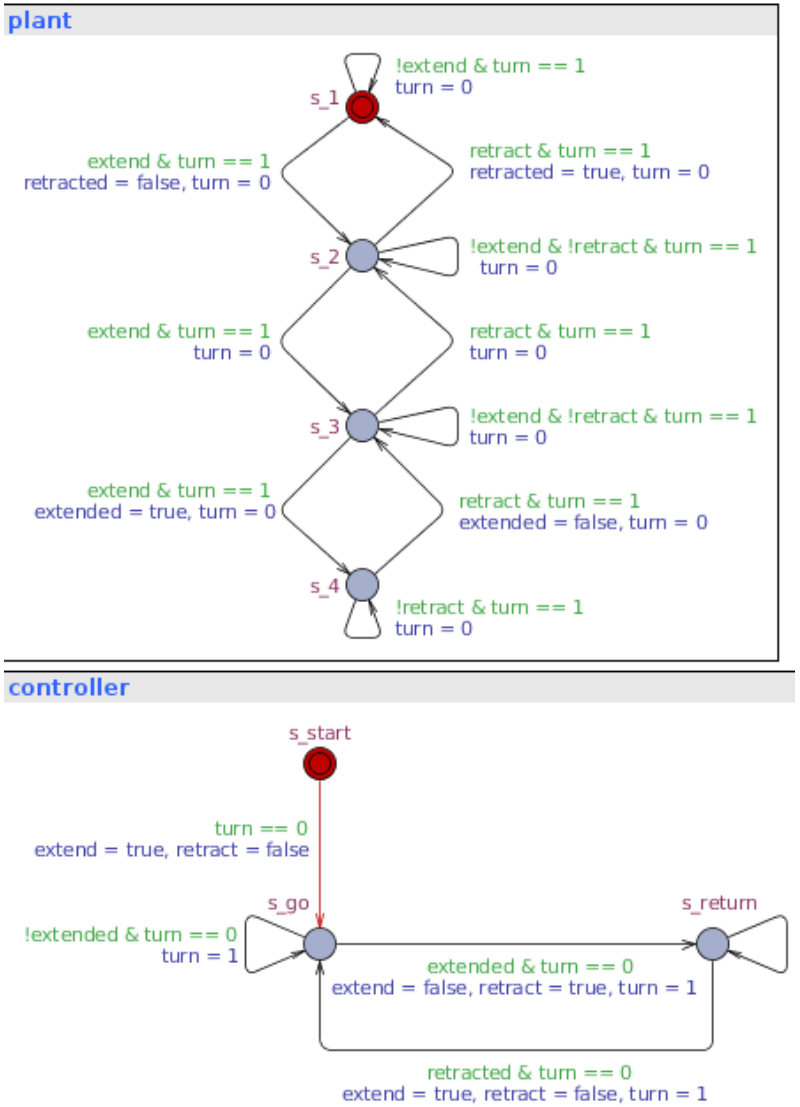
- $\mathrm{AP} = \{\text{“}p_i = j\text{”} \mid i = 1..3, j = 0..2\}$
- S: nodes of this graph
- $I \subset S = \{101\}$
- Note: in UPPAAL and NCES models there is always one initial state!
- $T \subset S \times S$: edges of the graph, e.g. $(002, 011)$, $(011, 002)$, ...
- $L : S \to 2^{\mathrm{AP}}$: token assignments (markings) in each state, e.g. $L(020) = \{\text{“}p_0 = 0\text{”}, \text{“}p_1 = 2\text{”}, \text{“}p_3 = 0\text{”}\}$

- **Specifications** can be interpreted as predicates over Kripke structures

Kripke structures / state graphs for UPPAAL models?

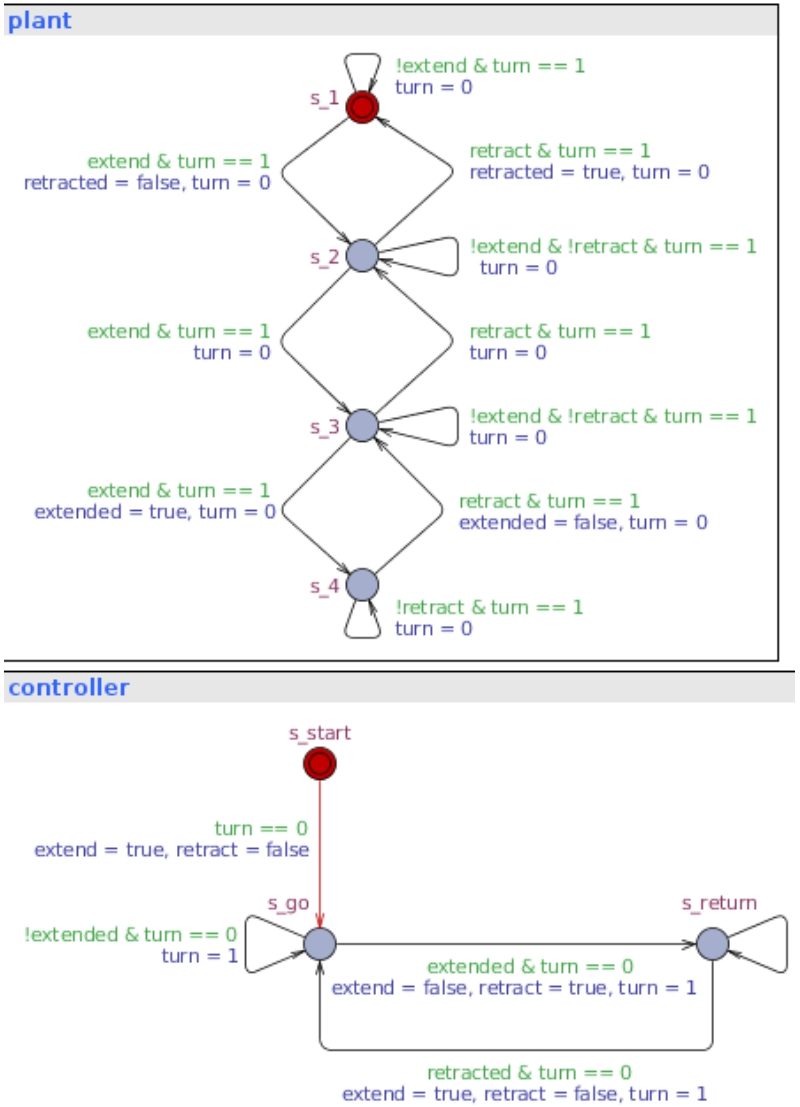


Kripke structures / state graphs for UPPAAL models?



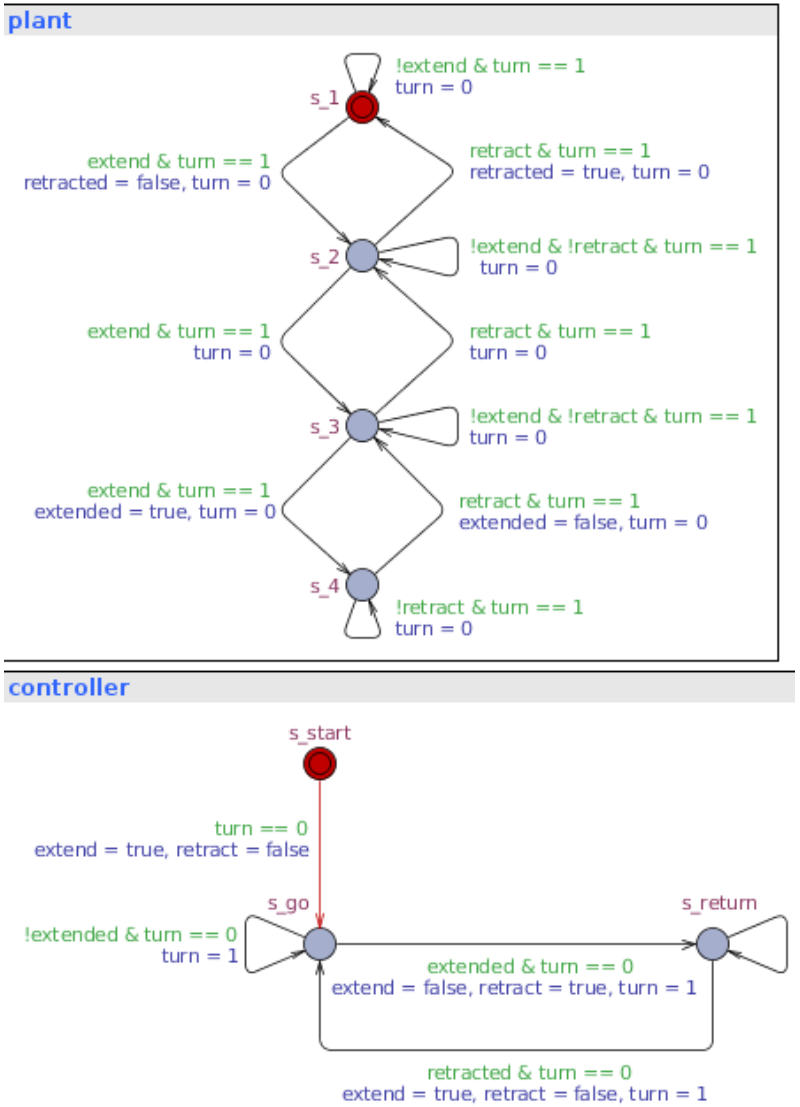
- AP: whether a state machine is in a certain state, whether a variable has a certain value
- S:

Kripke structures / state graphs for UPPAAL models?



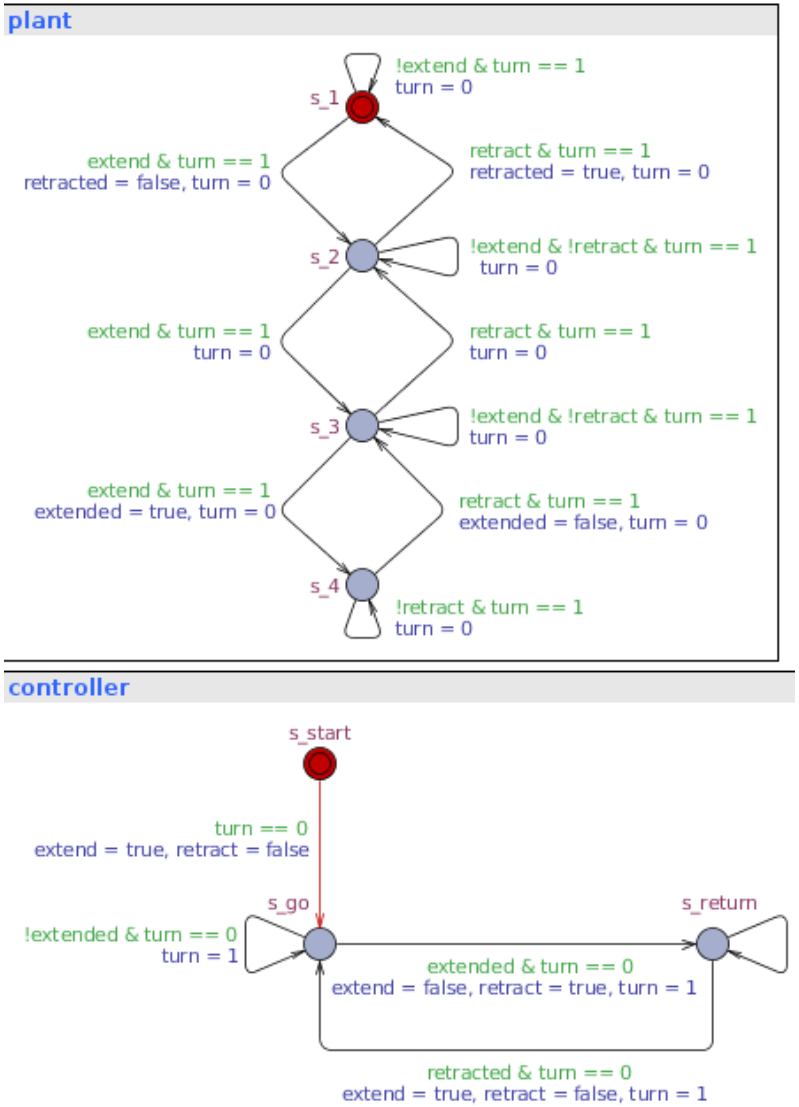
- AP: whether a state machine is in a certain state, whether a variable has a certain value
- S: $(s_1, ..., s_k, v_1, ..., v_m) \in S$ if it is a reachable combination of states and variable values
- $I \subset S$:

Kripke structures / state graphs for UPPAAL models?



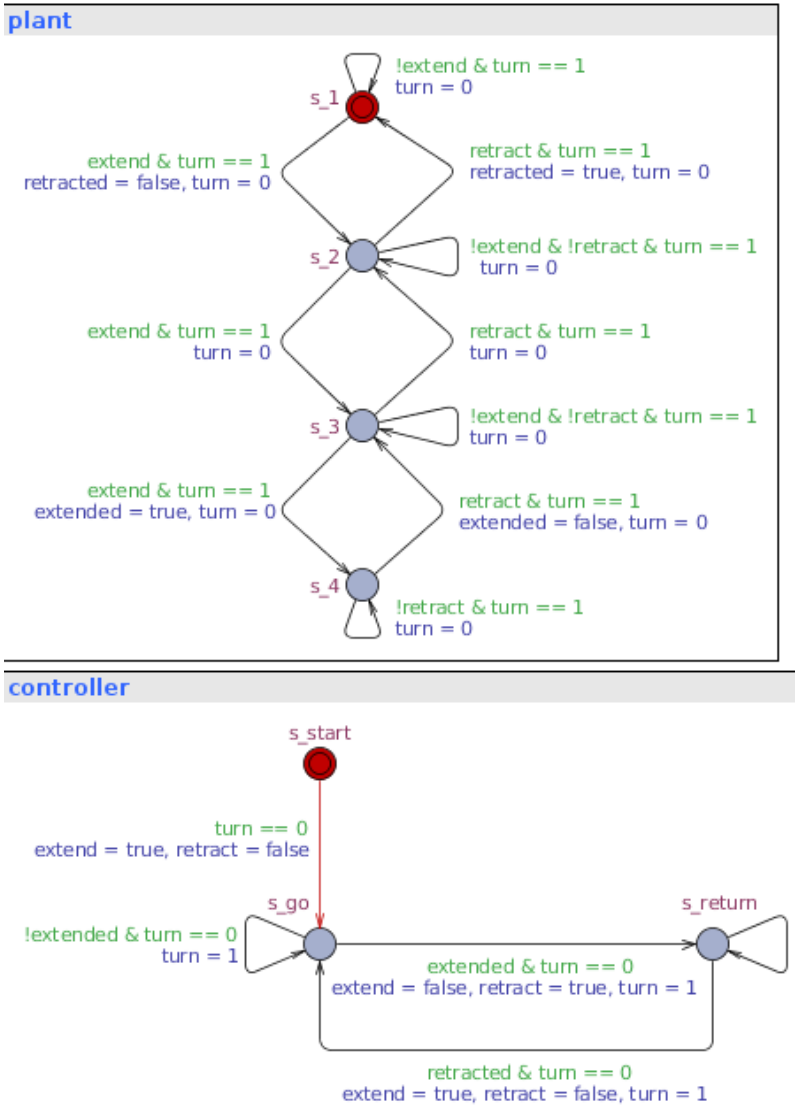
- AP: whether a state machine is in a certain state, whether a variable has a certain value
- S: $(s_1, \ldots, s_k, v_1, \ldots, v_m) \in S$ if it is a reachable combination of states and variable values
- $I \subset S$: single initial state $(s_{01}, \ldots, s_{0k}, v_{01}, \ldots, v_{0m})$ composed of initial individual states and variable values
- $T \subset S \times S$:

Kripke structures / state graphs for UPPAAL models?



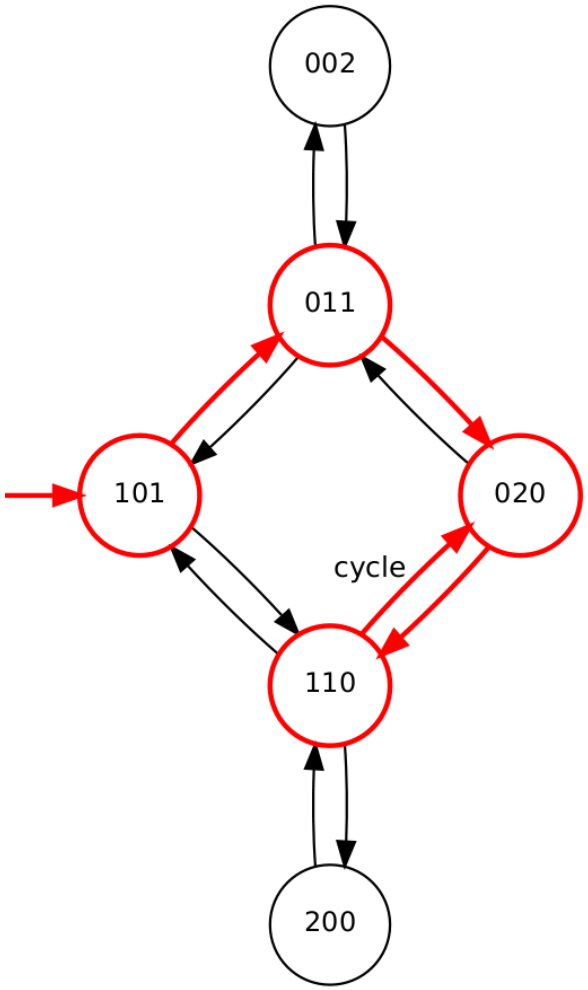
- AP: whether a state machine is in a certain state, whether a variable has a certain value
- S: $(s_1, \ldots, s_k, v_1, \ldots, v_m) \in S$ if it is a reachable combination of states and variable values
- $I \subset S$: single initial state $(s_{01}, \ldots, s_{0k}, v_{01}, \ldots, v_{0m})$ composed of initial individual states and variable values
- $T \subset S \times S$: valid state transitions
- $L : S \to 2^{\mathrm{AP}}$:

Kripke structures / state graphs for UPPAAL models?



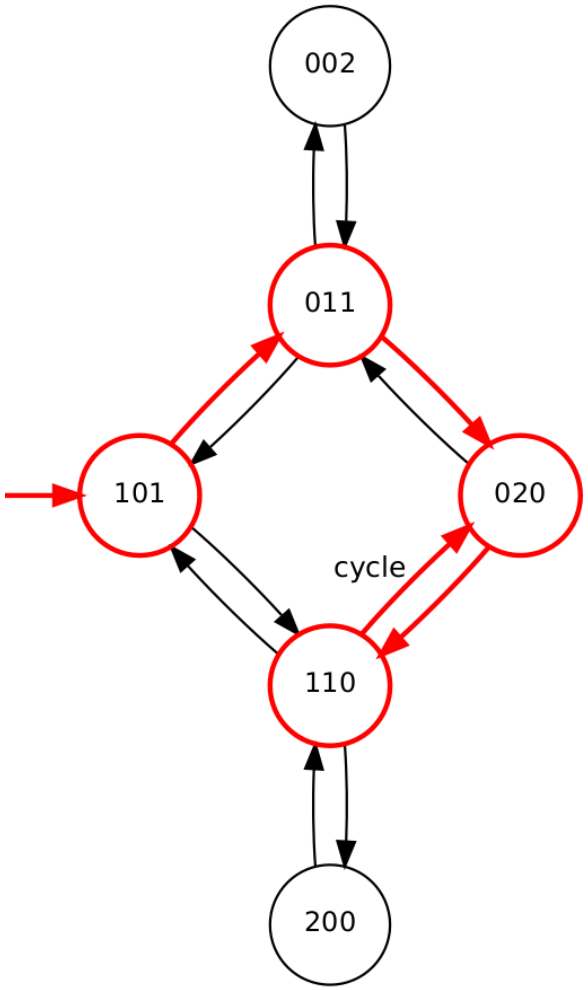
- AP: whether a state machine is in a certain state, whether a variable has a certain value
- S: $(s_1, ..., s_k, v_1, ..., v_m) \in S$ if it is a reachable combination of states and variable values
- $I \subset S$: single initial state $(s_{01}, ..., s_{0k}, v_{01}, ..., v_{0m})$ composed of initial individual states and variable values
- $T \subset S \times S$: valid state transitions
- $L : S \to 2^{AP}$: individual states and variable values
- Note: we ignore timed capabilities of UPPAAL by now

System behaviors are paths in Kripke structures

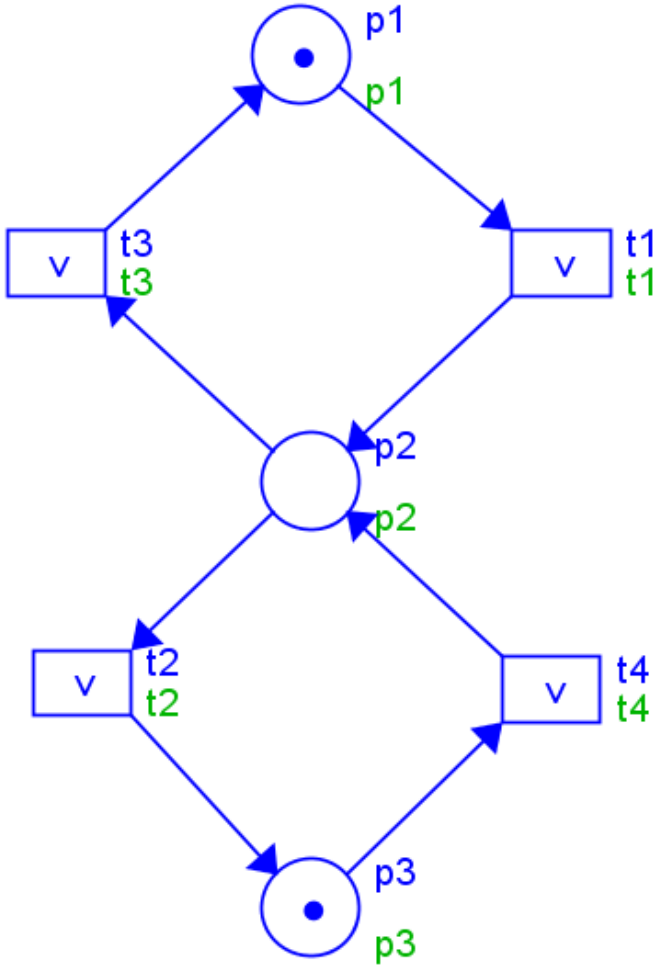


Infinite paths are common in formal verification. This is the reason why deadlocks are undesirable

System behaviors are paths in Kripke structures



Infinite paths are common in formal verification. This is the reason why deadlocks are undesirable



What happens in terms of the original system?

Single behavior view

- Assume that now we have only two atomic propositions: p and q
- All possible behaviors are infinite sequences over $2^{\{p,q\}}$
- Example: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$

Single behavior view

- Assume that now we have only two atomic propositions: p and q
- All possible behaviors are infinite sequences over $2^{\{p,q\}}$
- Example: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Boolean logic is able to characterize single elements of such sequences
- Can we somehow introduce predicates over infinite sequences of atomic propositions?
- For example, to formulate a specification: each p is followed by $\neg p$ on the next step (which is false for the example)

Linear temporal logic (LTL)

- Formal language which extends the usual propositional Boolean logic
- Variables: atomic propositions, e.g. p and q
- Usual Boolean operators are allowed, e.g. $p \to q$ (i.e. $\neg p \vee q$) is an LTL formula, but it refers to the **first element** of an infinite sequence

Linear temporal logic (LTL)

- Formal language which extends the usual propositional Boolean logic
- Variables: atomic propositions, e.g. p and q
- Usual Boolean operators are allowed, e.g. $p \to q$ (i.e. $\neg p \vee q$) is an LTL formula, but it refers to the **first element** of an infinite sequence

- **Temporal operators**

Linear temporal logic (LTL)

- Formal language which extends the usual propositional Boolean logic
- Variables: atomic propositions, e.g. p and q
- Usual Boolean operators are allowed, e.g. $p \to q$ (i.e. $\neg p \vee q$) is an LTL formula, but it refers to the **first element** of an infinite sequence

- **Temporal operators**
- **G**: **g**lobally (always), e.g. $\mathbf{G}(p \to q)$ means "in each element of the sequence, $p \to q$ holds"

Linear temporal logic (LTL)

- Formal language which extends the usual propositional Boolean logic
- Variables: atomic propositions, e.g. p and q
- Usual Boolean operators are allowed, e.g. $p \to q$ (i.e. $\neg p \vee q$) is an LTL formula, but it refers to the **first element** of an infinite sequence

- **Temporal operators**
- **G**: **g**lobally (always), e.g. $\mathbf{G}(p \to q)$ means "in each element of the sequence, $p \to q$ holds"
- **F**: in the **f**uture, e.g. $\mathbf{F}(p \to q)$ means "for some element of the sequence, $p \to q$ holds"

Linear temporal logic (LTL)

- Formal language which extends the usual propositional Boolean logic
- Variables: atomic propositions, e.g. p and q
- Usual Boolean operators are allowed, e.g. $p \to q$ (i.e. $\neg p \vee q$) is an LTL formula, but it refers to the **first element** of an infinite sequence

- **Temporal operators**
- **G**: **g**lobally (always), e.g. $\mathbf{G}(p \to q)$ means "in each element of the sequence, $p \to q$ holds"
- **F**: in the **f**uture, e.g. $\mathbf{F}(p \to q)$ means "for some element of the sequence, $p \to q$ holds"
- **X**: on the ne**x**t step, e.g. $\mathbf{X}(p \to q)$ means "$p \to q$ holds for the second element of the sequence"

Linear temporal logic (LTL)

- Formal language which extends the usual propositional Boolean logic
- Variables: atomic propositions, e.g. p and q
- Usual Boolean operators are allowed, e.g. $p \to q$ (i.e. $\neg p \vee q$) is an LTL formula, but it refers to the **first element** of an infinite sequence

- **Temporal operators**
- **G**: **g**lobally (always), e.g. $\mathbf{G}(p \to q)$ means "in each element of the sequence, $p \to q$ holds"
- **F**: in the **f**uture, e.g. $\mathbf{F}(p \to q)$ means "for some element of the sequence, $p \to q$ holds"
- **X**: on the ne**x**t step, e.g. $\mathbf{X}(p \to q)$ means "$p \to q$ holds for the second element of the sequence"
- **U**: **u**ntil (binary operator), e.g. $p\,\mathbf{U}\,q$ means "q must happen at some step, and the sequence must satisfy p until (non-inclusive) q happens"

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$ – path 2
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$ – path 2
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$ – paths 1, 3
- $f_3 = p\, \mathbf{U}(\neg p \wedge \neg q)$

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$ – path 2
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$ – paths 1, 3
- $f_3 = p\, \mathbf{U}(\neg p \wedge \neg q)$ – paths 1, 3

- Temporal operators can be applied to arbitrary LTL formulae!
- $f_4 = \mathbf{X}\,\mathbf{X}\,\mathbf{X}\,\mathbf{X}\, p$

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$ – path 2
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$ – paths 1, 3
- $f_3 = p\, \mathbf{U}(\neg p \wedge \neg q)$ – paths 1, 3

- Temporal operators can be applied to arbitrary LTL formulae!
- $f_4 = \mathbf{X}\,\mathbf{X}\,\mathbf{X}\,\mathbf{X}\, p$ ("on the fifth step") – paths 1, 2, 3
- $f_5 = \mathbf{F}\,\mathbf{G}(p \wedge q)$

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$ – path 2
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$ – paths 1, 3
- $f_3 = p\, \mathbf{U}(\neg p \wedge \neg q)$ – paths 1, 3

- Temporal operators can be applied to arbitrary LTL formulae!
- $f_4 = \mathbf{X}\,\mathbf{X}\,\mathbf{X}\,\mathbf{X}\, p$ ("on the fifth step") – paths 1, 2, 3
- $f_5 = \mathbf{F}\,\mathbf{G}(p \wedge q)$ ("globally from some point") – path 2
- $f_6 = \mathbf{G}\,\mathbf{F}(p \wedge q)$

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$ – path 2
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$ – paths 1, 3
- $f_3 = p\, \mathbf{U}(\neg p \wedge \neg q)$ – paths 1, 3

- Temporal operators can be applied to arbitrary LTL formulae!
- $f_4 = \mathbf{X}\,\mathbf{X}\,\mathbf{X}\,\mathbf{X}\, p$ ("on the fifth step") – paths 1, 2, 3
- $f_5 = \mathbf{F}\,\mathbf{G}(p \wedge q)$ ("globally from some point") – path 2
- $f_6 = \mathbf{G}\,\mathbf{F}(p \wedge q)$ ("infinitely often") – paths 1, 2, 3
- $f_7 = \mathbf{G}(p \rightarrow \mathbf{X}\, q)$

Examples of LTL formulae

- Path 1: $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$
- Path 2: $\text{cycle}(\{p, q\})$
- Path 3: $\{\}, \text{cycle}(\{p\}, \{p, q\}, \{q\})$

- $f_1 = \mathbf{G}\, p$ – path 2
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$ – paths 1, 3
- $f_3 = p\, \mathbf{U}(\neg p \wedge \neg q)$ – paths 1, 3

- Temporal operators can be applied to arbitrary LTL formulae!
- $f_4 = \mathbf{X}\,\mathbf{X}\,\mathbf{X}\,\mathbf{X}\, p$ ("on the fifth step") – paths 1, 2, 3
- $f_5 = \mathbf{F}\,\mathbf{G}(p \wedge q)$ ("globally from some point") – path 2
- $f_6 = \mathbf{G}\,\mathbf{F}(p \wedge q)$ ("infinitely often") – paths 1, 2, 3
- $f_7 = \mathbf{G}(p \rightarrow \mathbf{X}\, q)$ ("p is always followed by q") – paths 2, 3

Path visualization tool

- Download from `https://github.com/igor-buzhinsky/nusmv_counterexample_visualizer`
- The tool supports interpreting counterexamples produced by NuSMV (will be covered later in the course), but can also be used to **structurally explain** LTL formula values on user-specified paths
- **Important atomic propositions** are highlighted

- Variable view: suppose that the state of the formal model is composed on a number of variables, either Boolean or integer
- Boolean variables can be interpreted as atomic propositions right away
- Statements over integer variables (e.g. comparisons like $x > 5$) can also be interepreted as atomic propositions

Path visualization tool: example

- Visualize $\mathbf{G}(p \to \mathbf{X}\, q)$ on path $\{p, q\}, \{p\}, \{\}, \text{cycle}(\{q\}, \{p, q\})$:

#	Value	LTL specification
7	TRUE	G F (p & q)
8	TRUE	G F (p & q)
9	TRUE	G F (p & q)
10	FALSE	G (p -> X q)
11	TRUE	G (p -> X q)
12	TRUE	G (p -> X q)

```
step 0  G((p -> X(q)))
prefix  ‖ ‖T     ‖ T‖‖‖
        ‖ ‖      ‖FF‖‖‖
        ‖ ‖FFFFFFFFF‖‖
        ‖FFFFFFFFFFF‖
step 1  G((p -> X(q)))
prefix  ‖ ‖T     ‖ F‖‖‖
        ‖ ‖      ‖FF‖‖‖
        ‖ ‖FFFFFFFFF‖‖
        ‖FFFFFFFFFFF‖
step 2  G((p -> X(q)))
prefix  ‖ ‖F     ‖ F‖‖‖
        ‖ ‖      ‖TT‖‖‖
```

Step	p	q
0 prefix	TRUE	TRUE
1 prefix	TRUE	FALSE
2 prefix	FALSE	FALSE
3 loop	FALSE	TRUE
4 loop	TRUE	TRUE

Path visualization tool: input format

```
-- specification  G (p -> X q)
  -> new state <-
    p = TRUE
    q = TRUE
  -> new state <-
    p = TRUE
    q = FALSE
  -> new state <-
    p = FALSE
    q = FALSE
  -- Loop starts here
  -> new state <-
    p = FALSE
    q = TRUE
  -> new state <-
    p = TRUE
    q = TRUE
```

LTL: simplification and equivalence rules

- $\mathbf{G}\,\mathbf{G}\,f = \mathbf{G}\,f$
- $\mathbf{F}\,\mathbf{F}\,f = \mathbf{F}\,f$

LTL: simplification and equivalence rules

- $\mathbf{G}\,\mathbf{G}\,f = \mathbf{G}\,f$
- $\mathbf{F}\,\mathbf{F}\,f = \mathbf{F}\,f$

- $\mathbf{G}\,\mathbf{X}\,f = \mathbf{X}\,\mathbf{G}\,f$
- $\mathbf{F}\,\mathbf{X}\,f = \mathbf{X}\,\mathbf{F}\,f$

LTL: simplification and equivalence rules

- $\mathbf{G}\,\mathbf{G}\,f = \mathbf{G}\,f$
- $\mathbf{F}\,\mathbf{F}\,f = \mathbf{F}\,f$

- $\mathbf{G}\,\mathbf{X}\,f = \mathbf{X}\,\mathbf{G}\,f$
- $\mathbf{F}\,\mathbf{X}\,f = \mathbf{X}\,\mathbf{F}\,f$

- $\neg\,\mathbf{G}(f) = \mathbf{F}(\neg f)$
- $\neg\,\mathbf{F}(f) = \mathbf{G}(\neg f)$

LTL verification: definition

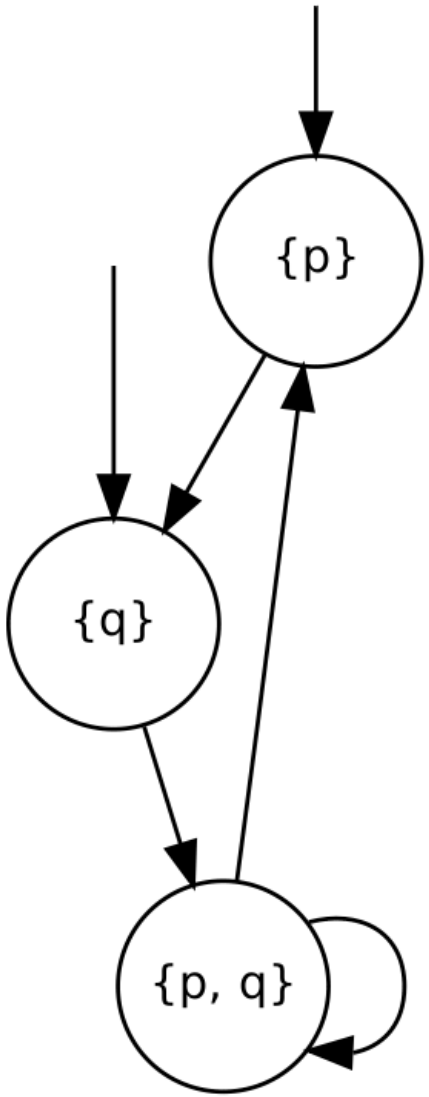
- Kripke structure M satisfies LTL formula f (written: $M \models f$), if **all paths** in M which **start in** M**'s initial states** satisfy f

LTL verification: definition

- Kripke structure M satisfies LTL formula f (written: $M \models f$), if **all paths** in M which **start in** M**'s initial states** satisfy f

Quiz: which of these LTL formulae are satisfied by the KS on the right? Why?

- $f_1 = \mathbf{G}\, p$
- $f_2 = \mathbf{F}(\neg p \wedge \neg q)$
- $f_3 = p\, \mathbf{U}(\neg p \wedge \neg q)$
- $f_4 = \mathbf{X}\,\mathbf{X}\,\mathbf{X}\,\mathbf{X}\, p$
- $f_5 = \mathbf{F}\,\mathbf{G}(p \wedge q)$
- $f_6 = \mathbf{G}\,\mathbf{F}(p \wedge q)$
- $f_7 = \mathbf{G}(p \rightarrow \mathbf{X}\, q)$



Quiz answers

Quiz answers

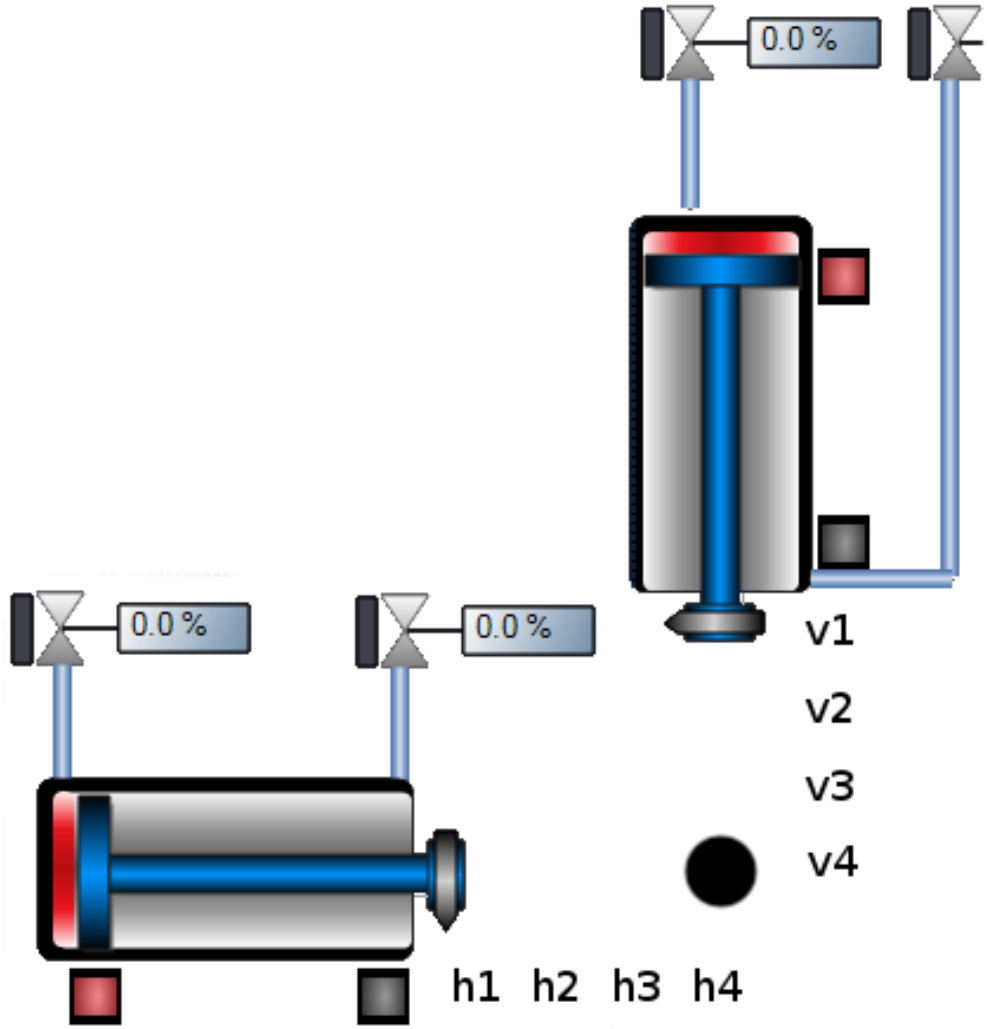
- Only $f_6 = \mathbf{G}\,\mathbf{F}(p \wedge q)$

LTL verification: **model checking** algorithm

- We wish to check whether f holds for Kripke structure M
- Automata-theoretic approach
- $\neg f$ is converted to a so-called **Büchi automaton**, which is an acceptor over infinite words which satisfy $\neg f$
- M is composed with this automaton
- If the composition accepts at least one infinite word, then this word satisfies $\neg f$ and belongs to M, so f is false, and the obtained word is a **counterexample**
- Otherwise, f is true

- We won't go into details

Two cylinders system



- The extension of each cylinder is discretized into four intervals
- When both cylinders share interval 4, they collide
- A workpiece can be placed into the shared interval
- If a cylinder reaches interval 4 and there is a workpiece, it is pushed

LTL specification for the two cylinders system: plant model

- Atomic propositions: h_1, h_2, h_3, h_4 (displacements of the horizonal cylinder), v_1, v_2, v_3, v_4 (displacements of the vertical cylinder), w (workpiece is present)
- Cylinder has a position:

LTL specification for the two cylinders system: plant model

- Atomic propositions: h_1, h_2, h_3, h_4 (displacements of the horizonal cylinder), v_1, v_2, v_3, v_4 (displacements of the vertical cylinder), w (workpiece is present)

- Cylinder has a position: $\mathbf{G}(h_1 \vee h_2 \vee h_3 \vee h_4)$, $\mathbf{G}(v_1 \vee v_2 \vee v_3 \vee v_4)$
- Cylinder can't have more than one position:

LTL specification for the two cylinders system: plant model

- Atomic propositions: h_1, h_2, h_3, h_4 (displacements of the horizonal cylinder), v_1, v_2, v_3, v_4 (displacements of the vertical cylinder), w (workpiece is present)

- Cylinder has a position: $\mathbf{G}(h_1 \vee h_2 \vee h_3 \vee h_4)$, $\mathbf{G}(v_1 \vee v_2 \vee v_3 \vee v_4)$
- Cylinder can't have more than one position:
  $\mathbf{G}(\neg(h_1 \wedge h_2) \wedge \neg(h_1 \wedge h_3) \wedge \neg(h_1 \wedge h_4) \wedge \neg(h_2 \wedge h_3) \wedge \neg(h_2 \wedge h_4) \wedge \neg(h_3 \wedge h_4))$,
  $\mathbf{G}(\neg(v_1 \wedge v_2) \wedge \neg(v_1 \wedge v_3) \wedge \neg(v_1 \wedge v_4) \wedge \neg(v_2 \wedge v_3) \wedge \neg(v_2 \wedge v_4) \wedge \neg(v_3 \wedge v_4))$
- If a cylinder is fully extended, then there is no workpiece:

LTL specification for the two cylinders system: plant model

- Atomic propositions: h_1, h_2, h_3, h_4 (displacements of the horizonal cylinder), v_1, v_2, v_3, v_4 (displacements of the vertical cylinder), w (workpiece is present)

- Cylinder has a position: $\mathbf{G}(h_1 \vee h_2 \vee h_3 \vee h_4)$, $\mathbf{G}(v_1 \vee v_2 \vee v_3 \vee v_4)$
- Cylinder can't have more than one position:
  $\mathbf{G}(\neg(h_1 \wedge h_2) \wedge \neg(h_1 \wedge h_3) \wedge \neg(h_1 \wedge h_4) \wedge \neg(h_2 \wedge h_3) \wedge \neg(h_2 \wedge h_4) \wedge \neg(h_3 \wedge h_4))$,
  $\mathbf{G}(\neg(v_1 \wedge v_2) \wedge \neg(v_1 \wedge v_3) \wedge \neg(v_1 \wedge v_4) \wedge \neg(v_2 \wedge v_3) \wedge \neg(v_2 \wedge v_4) \wedge \neg(v_3 \wedge v_4))$
- If a cylinder is fully extended, then there is no workpiece:
  $\mathbf{G}(h_4 \vee v_4 \rightarrow \neg w)$
- ...

- Such specifications can help "debug" the plant model

LTL specification for the two cylinders system: requirements for the controller

- We won't use any additional atomic propositions: all we need can be specified in terms of the plant!

LTL specification for the two cylinders system: requirements for the controller

- We won't use any additional atomic propositions: all we need can be specified in terms of the plant!
- Cylinders do not collide:

LTL specification for the two cylinders system: requirements for the controller

- We won't use any additional atomic propositions: all we need can be specified in terms of the plant!
- Cylinders do not collide: $\mathbf{G}\,\neg(h_4 \wedge v_4)$

LTL specification for the two cylinders system: requirements for the controller

- We won't use any additional atomic propositions: all we need can be specified in terms of the plant!

- Cylinders do not collide: $\mathbf{G}\,\neg(h_4 \wedge v_4)$
- When a workpiece appears, it must be eventually pushed away:

LTL specification for the two cylinders system: requirements for the controller

- We won't use any additional atomic propositions: all we need can be specified in terms of the plant!

- Cylinders do not collide: $\mathbf{G}\,\neg(h_4 \wedge v_4)$
- When a workpiece appears, it must be eventually pushed away: $\mathbf{G}(w \rightarrow \mathbf{F}\,\neg w)$

LTL specification for the two cylinders system: requirements for the controller

- We won't use any additional atomic propositions: all we need can be specified in terms of the plant!

- Cylinders do not collide: $\mathbf{G}\,\neg(h_4 \wedge v_4)$
- When a workpiece appears, it must be eventually pushed away: $\mathbf{G}(w \rightarrow \mathbf{F}\,\neg w)$
- Cylinders iterate (each new workpiece is pushed by a different cylinder):

LTL specification for the two cylinders system: requirements for the controller

- We won't use any additional atomic propositions: all we need can be specified in terms of the plant!

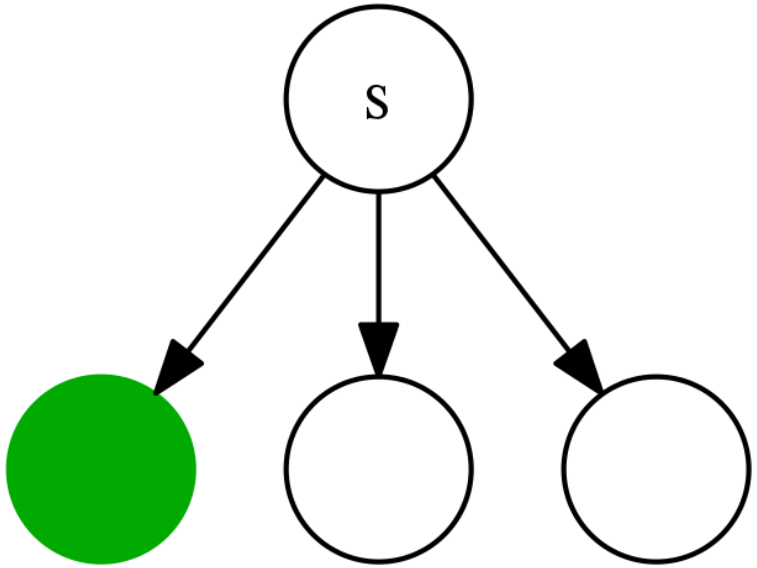
- Cylinders do not collide: $\mathbf{G}\,\neg(h_4 \wedge v_4)$
- When a workpiece appears, it must be eventually pushed away: $\mathbf{G}(w \rightarrow \mathbf{F}\,\neg w)$
- Cylinders iterate (each new workpiece is pushed by a different cylinder): $\mathbf{G}((h_4 \wedge (\mathbf{X}\,\neg h_4) \wedge \mathbf{F}\,w) \rightarrow \mathbf{X}((\neg w \wedge \neg v_4 \wedge \neg h_4)\,\mathbf{U}(w \wedge (w\,\mathbf{U}(v_4 \wedge \neg h_4)))))$, and the same for the other cylinder

Computation tree logic (CTL)

- In LTL, there is always an implicit quantification over all paths starting in initial states
- In **CTL**, all temporal operators are annotated with quantifiers

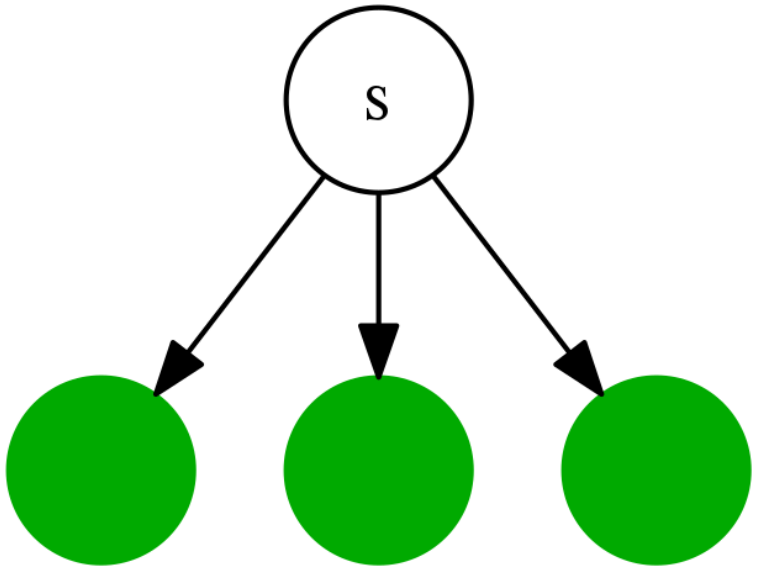
- CTL formulae characterize not infinite sequences, but rather states of the Kripke structure
- A Kripke structure satisfies a CTL formula, if **all its initial states** satisfy this formula
- Let s be a state of the KS, then $s \models f$ means s satisfies f

CTL: temporal operator **EX**



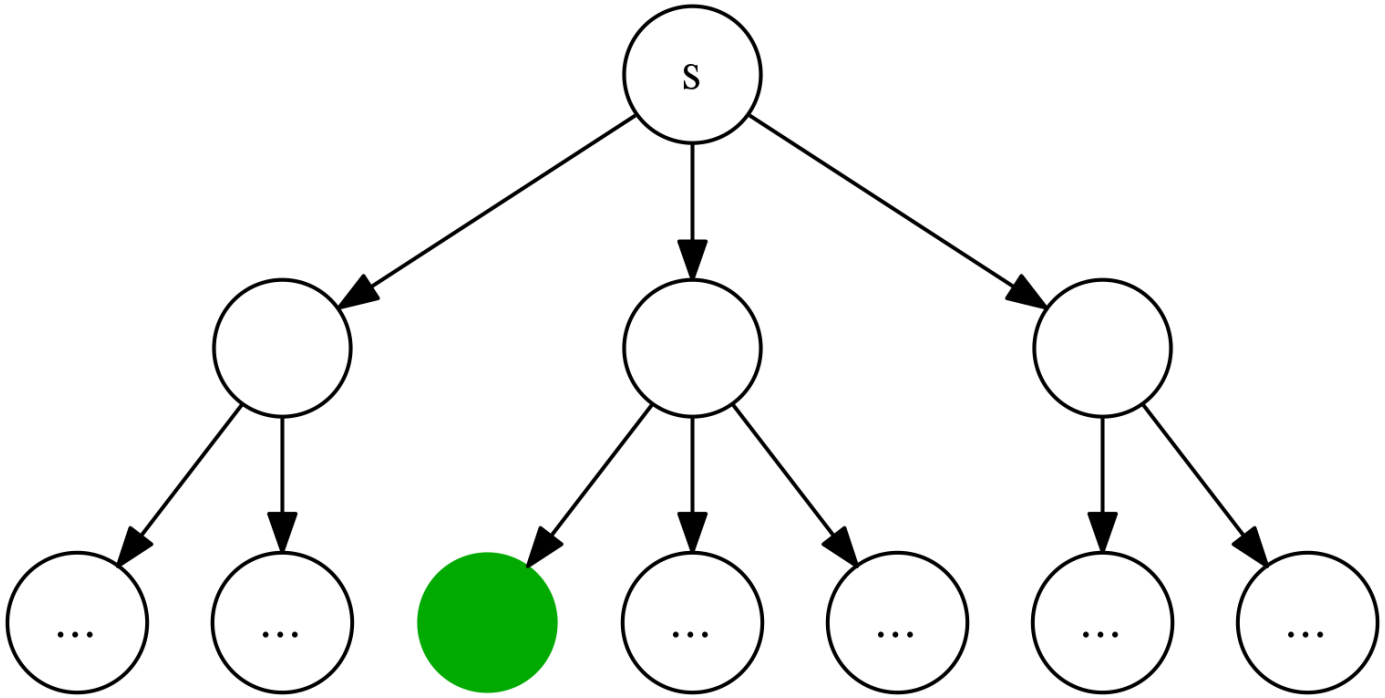
- $s \models \mathbf{EX}(f)$ ("exists next", not supported by UPPAAL): there is a successor of s where f holds

CTL: temporal operator **AX**



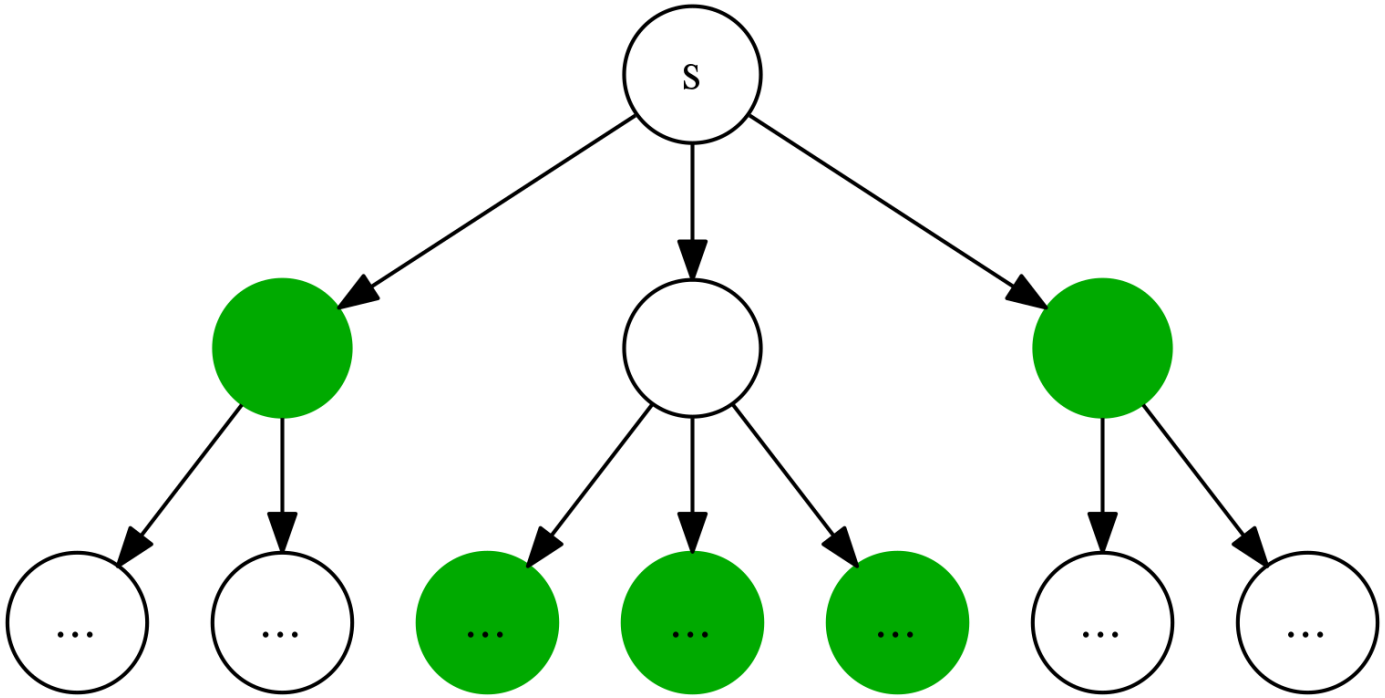
- $s \models \mathbf{AX}(f)$ ("for all next", not supported by UPPAAL): in all successors of s, f holds

CTL: temporal operator **EF**



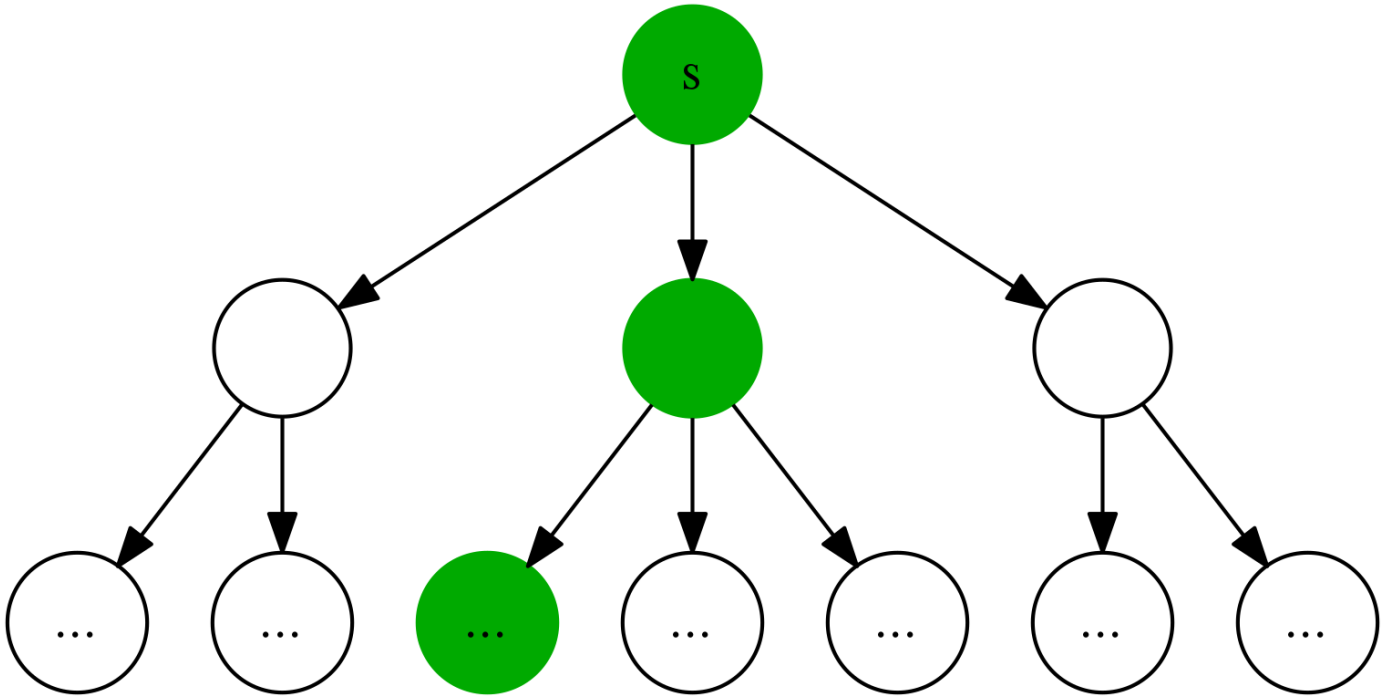
- $s \models \mathbf{EF}(f)$ ("exists in the future", E<> in UPPAAL): there exists a path starting in s such that f becomes valid at some point of this path

CTL: temporal operator **AF**



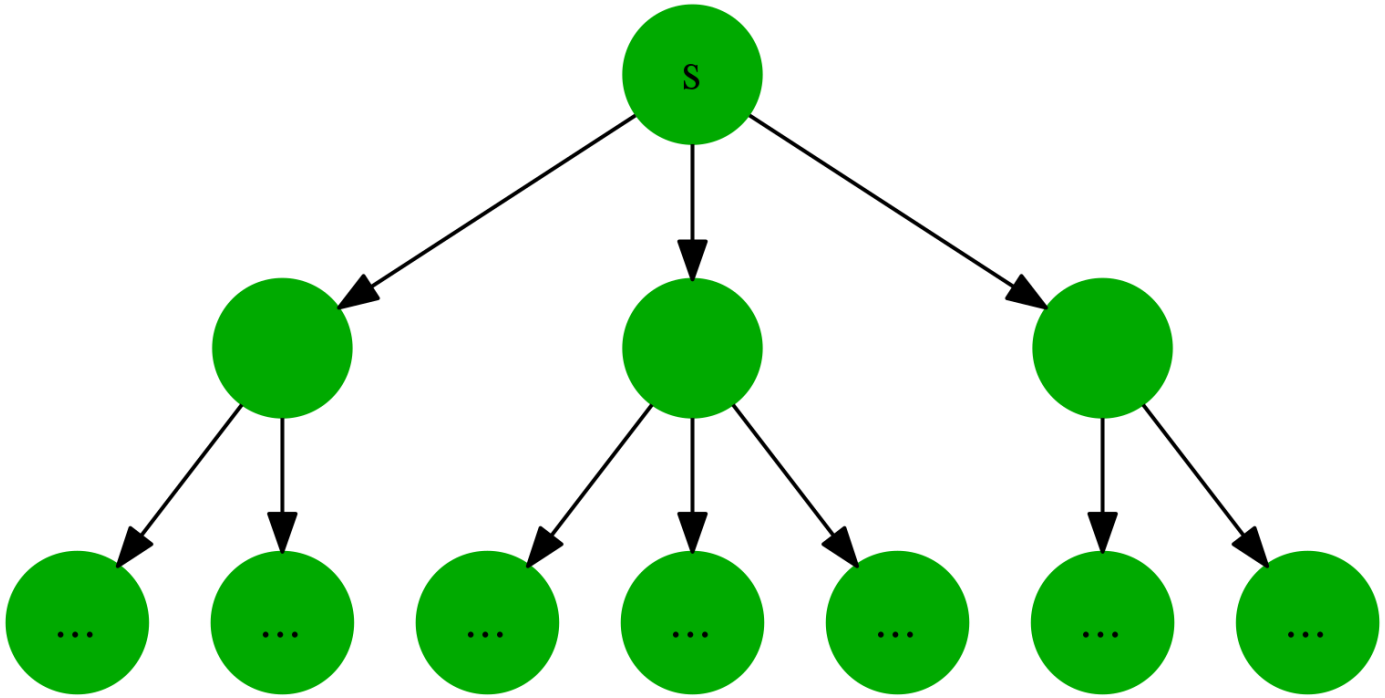
- $s \models \mathbf{AF}(f)$ ("for all in the future", A<> in UPPAAL): for all possible paths starting in s, f becomes true at some point

CTL: temporal operator **EG**



- $s \models \mathbf{EG}(f)$ ("exists globally", E[] in UPPAAL): there exists a path starting in s such that f holds at every state along this path

CTL: temporal operator **AG**



- $s \models \mathbf{AG}(f)$ ("for all globally", A[] in UPPAAL): for all possible paths starting in s, f is always true

CTL: temporal operators **EU** and **AU**

- $s \models f$ **EU** g ("exists until", not supported by UPPAAL): there exists a path starting in s such that f holds until (non-inclusive) g, and g eventually happens
- $s \models f$ **AU** g ("for all until", not supported by UPPAAL): for all possible paths starting in s, f holds until (non-inclusive) g, and g eventually happens

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$ – CTL

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$ – CTL
- $\mathbf{EX}\,\neg\,\mathbf{AG}\, q$

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$ – CTL
- $\mathbf{EX}\,\neg\,\mathbf{AG}\, q$ – CTL

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$ – CTL
- $\mathbf{EX}\,\neg\,\mathbf{AG}\, q$ – CTL
- $\mathbf{G}(p \rightarrow \mathbf{X}\,\mathbf{X}\,\mathbf{F}\, q)$

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$ – CTL
- $\mathbf{EX}\,\neg\,\mathbf{AG}\, q$ – CTL
- $\mathbf{G}(p \rightarrow \mathbf{X}\,\mathbf{X}\,\mathbf{F}\, q)$ – LTL

Are these formulae syntactically correct in LTL or CTL?

- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$ – CTL
- $\mathbf{EX}\,\neg\,\mathbf{AG}\, q$ – CTL
- $\mathbf{G}(p \rightarrow \mathbf{X}\,\mathbf{X}\,\mathbf{F}\, q)$ – LTL
- $(\mathbf{AX}\, p)\,\mathbf{U}(\mathbf{EF}\,\neg p)$

Are these formulae syntactically correct in LTL or CTL?

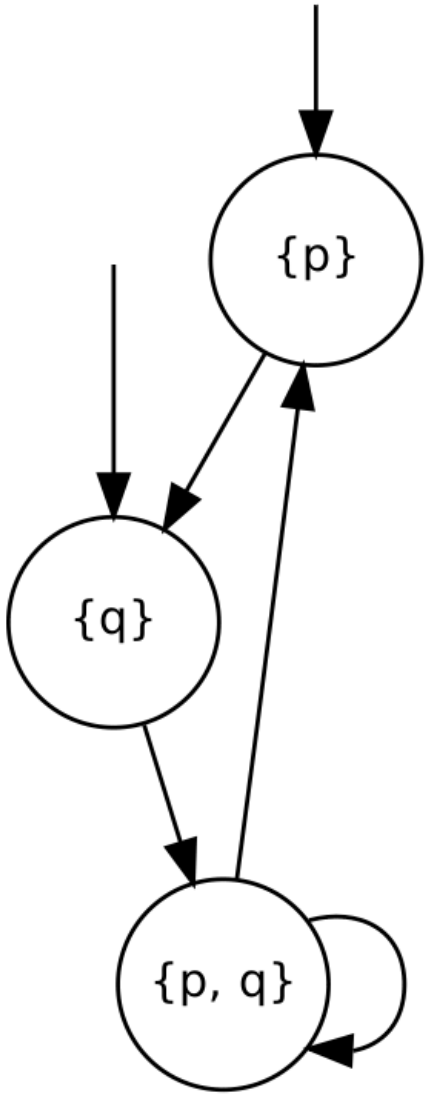
- $p \wedge \neg q$ – both LTL and CTL
- $\mathbf{AX}(p \rightarrow \mathbf{F}\, q)$ – incorrect
- $\mathbf{F}\,\mathbf{X}\,\mathbf{AG}\, q$ – incorrect
- $\mathbf{EX}\,\mathbf{AG}\, q$ – CTL
- $\mathbf{EX}\,\neg\,\mathbf{AG}\, q$ – CTL
- $\mathbf{G}(p \rightarrow \mathbf{X}\,\mathbf{X}\,\mathbf{F}\, q)$ – LTL
- $(\mathbf{AX}\, p)\,\mathbf{U}(\mathbf{EF}\,\neg p)$ – incorrect

CTL verification: example

- KS satisfies the CTL formula iff all its initial states satisfy it

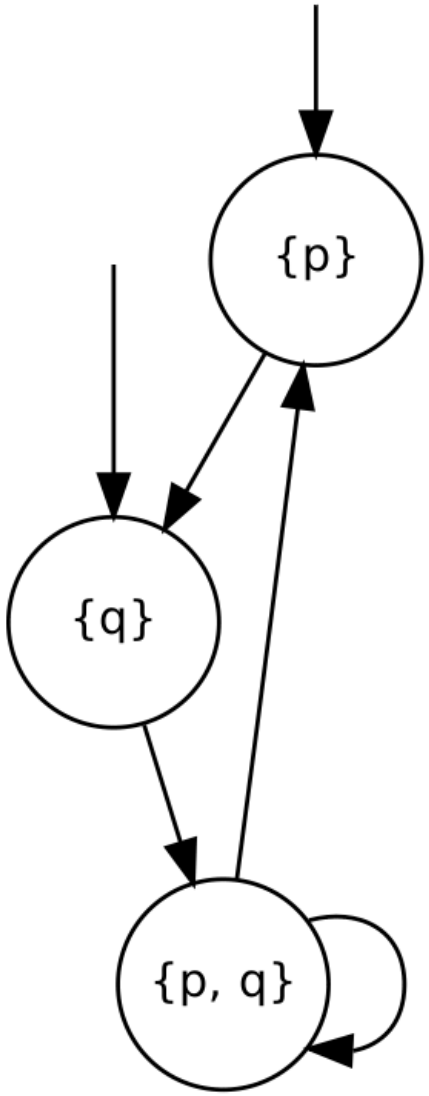
CTL verification: example

- KS satisfies the CTL formula iff all its initial states satisfy it
- Which of these CTL formulae are satisfied by the KS on the right? Why?
- $f_1 = \mathbf{AG}\, p$
- $f_2 = \mathbf{AG}(p \vee q)$
- $f_3 = \mathbf{AF}(p \wedge q)$
- $f_4 = \mathbf{EF}(\neg p \wedge \neg q)$
- $f_5 = \mathbf{AX}\,\mathbf{AX}\,\mathbf{AX}\,\mathbf{AX}\, p$
- $f_6 = \mathbf{EF}\,\mathbf{EG}(p \wedge q)$
- $f_7 = \mathbf{EG}\,\mathbf{EF}(p \wedge q)$
- $f_8 = \mathbf{AG}(p \rightarrow \mathbf{AX}\, q)$



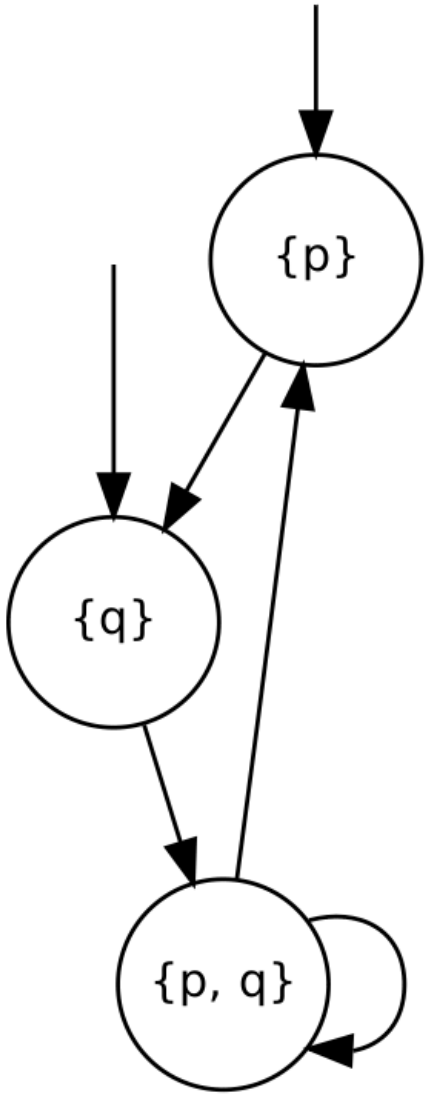
CTL verification: example

- KS satisfies the CTL formula iff all its initial states satisfy it
- Which of these CTL formulae are satisfied by the KS on the right? Why?
- $f_1 = \mathbf{AG}\, p$
- $f_2 = \mathbf{AG}(p \vee q)$
- $f_3 = \mathbf{AF}(p \wedge q)$
- $f_4 = \mathbf{EF}(\neg p \wedge \neg q)$
- $f_5 = \mathbf{AX}\,\mathbf{AX}\,\mathbf{AX}\,\mathbf{AX}\, p$
- $f_6 = \mathbf{EF}\,\mathbf{EG}(p \wedge q)$
- $f_7 = \mathbf{EG}\,\mathbf{EF}(p \wedge q)$
- $f_8 = \mathbf{AG}(p \rightarrow \mathbf{AX}\, q)$
- Answer:



CTL verification: example

- KS satisfies the CTL formula iff all its initial states satisfy it
- Which of these CTL formulae are satisfied by the KS on the right? Why?
- $f_1 = \mathbf{AG}\, p$
- $f_2 = \mathbf{AG}(p \vee q)$
- $f_3 = \mathbf{AF}(p \wedge q)$
- $f_4 = \mathbf{EF}(\neg p \wedge \neg q)$
- $f_5 = \mathbf{AX}\,\mathbf{AX}\,\mathbf{AX}\,\mathbf{AX}\, p$
- $f_6 = \mathbf{EF}\,\mathbf{EG}(p \wedge q)$
- $f_7 = \mathbf{EG}\,\mathbf{EF}(p \wedge q)$
- $f_8 = \mathbf{AG}(p \rightarrow \mathbf{AX}\, q)$
- Answer: f_2, f_3, f_6, f_7



CTL verification: model checking algorithms

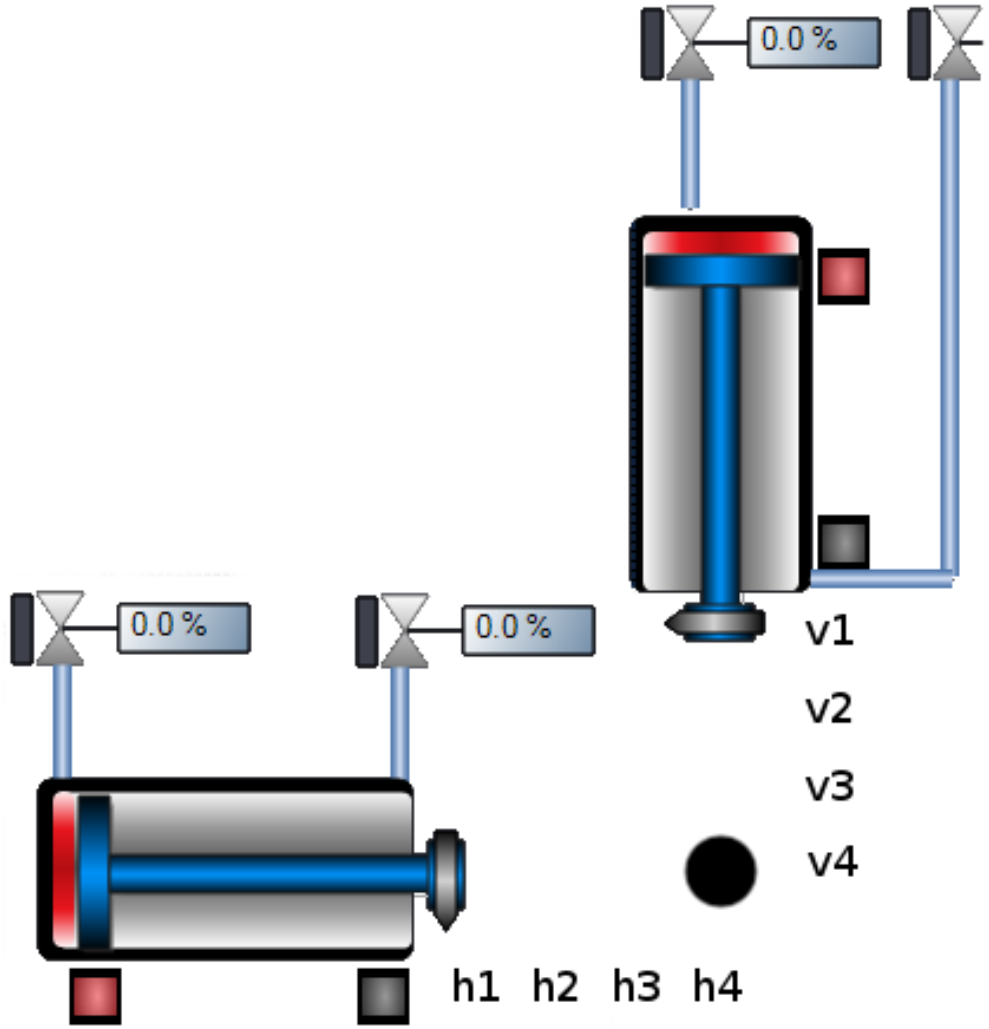
Graph theory approach

- If f is just a Boolean formula, then it is trivial to check whether it holds in a desired state
- **AX** f holds in s, if we previously found that f holds for each in all its successors
- Similar (sometimes more complex) ideas for other operators
- We won't go into details

Symbolic approach

- Implicit representation of states via Boolean formulae
- Allows partial mitigation of the **state explosion problem**
- The symbolic approach will be examined in more detail later in the course

CTL verification: two cylinders



- Atomic propositions: $h_1..h_4, v_1..v_4, w$

Quiz: specify the following properties in CTL:

- If a cylinder is fully extended, then there is no workpiece
- Cylinders do not collide
- When a workpiece appears, it must be eventually pushed away
- Cylinders iterate

Quiz answers

Quiz answers

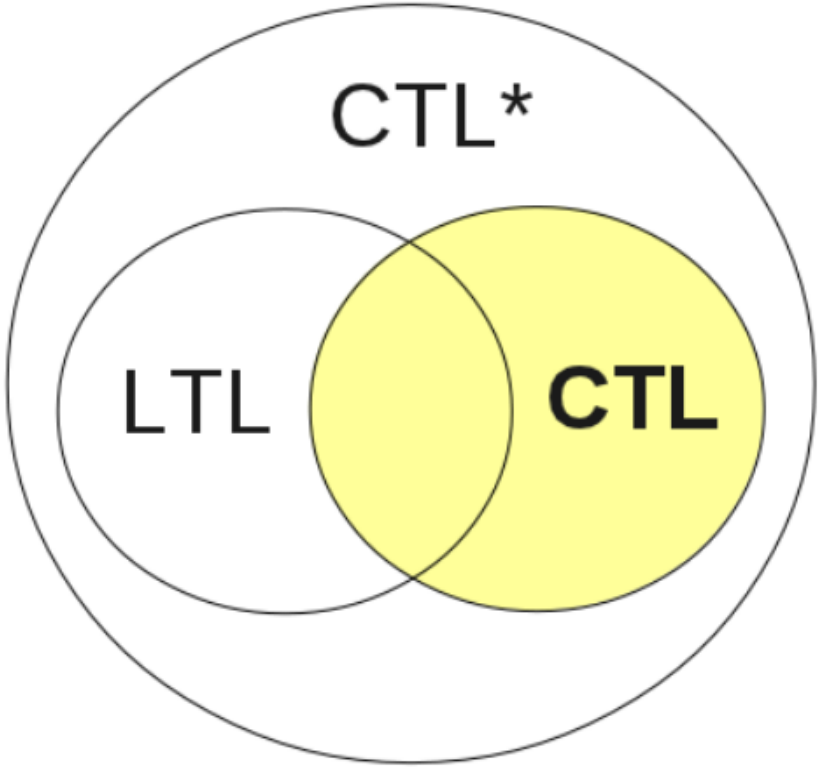
- If a cylinder is fully extended, then there is no workpiece: $\mathbf{AG}(h_4 \vee v_4 \rightarrow \neg w)$
- Cylinders do not collide: $\mathbf{AG}\,\neg(h_4 \wedge v_4)$
- When a workpiece appears, it must be eventually pushed away: $\mathbf{AG}(w \rightarrow \mathbf{AF}\,\neg w)$
- Cylinders iterate: ???

Common specification types ("patterns")

Name	LTL	CTL
Generality / Invariance	$\mathbf{G}\, f$	$\mathbf{AG}\, f$
Bounded response	$\mathbf{G}(p \to \mathbf{X}^n\, q)$	$\mathbf{AG}(p \to (\mathbf{AX})^n q)$
Unbounded response	$\mathbf{G}(p \to \mathbf{F}\, q)$	$\mathbf{AG}(p \to \mathbf{AF}\, q)$
Infinitely often	$\mathbf{G}\,\mathbf{F}\, p$	$\mathbf{AG}\,\mathbf{AF}\, p$

There are properties which cannot be expressed in both LTL and CTL

- **G** p/ **AG** p, **F** p/ **AF** p, **G F** p/ **AG AF** p – both LTL and CTL
- **EF** p – only CTL, but there is a workaround to check it in LTL!
- **F G** p – only LTL
- **AG EF** p – only CTL
- CTL* is a larger logic which allows combining quantified and unquantified temporal operators



Tool support

- UPPAAL: a subset of CTL, some LTL properties can be specified with automata
- NuSMV: both CTL and LTL
- SPIN: LTL

Literature

1. Baier, C., & Katoen, J. P. (2008). Principles of model checking. MIT press.
2. Clarke, E. M., Grumberg, O., & Peled, D. (1999). Model checking. MIT press.
3. Schneider, K. (2013). Verification of reactive systems: formal methods and algorithms. Springer Science & Business Media.