Model-based Testing and Model Checking for Safety-critical Hierarchical Systems

December 6, 2018

M.Sc. thesis in Computer Science by
Luigi Libero Lucio Starace

Università degli Studi di Napoli Federico II

This thesis work
An overview

- Verification of industrial controllers is the most critical and time-consuming phase in the development life-cycle.

This thesis work
An overview

- Verification of industrial controllers is the most critical and time-consuming phase in the development life-cycle.
- Techniques are sought to ease verification efforts and increase effectiveness.

This thesis work
An overview

- Verification of industrial controllers is the most critical and time-consuming phase in the development life-cycle.
- Techniques are sought to ease verification efforts and increase effectiveness.
- Extend previous research on Dynamic State Machines (DSTM):

This thesis work
An overview

- Verification of industrial controllers is the most critical and time-consuming phase in the development life-cycle.
- Techniques are sought to ease verification efforts and increase effectiveness.
- Extend previous research on Dynamic State Machines (DSTM):
  - a new test case generation procedure;

This thesis work
An overview

- Verification of industrial controllers is the most critical and time-consuming phase in the development life-cycle.
- Techniques are sought to ease verification efforts and increase effectiveness.
- Extend previous research on Dynamic State Machines (DSTM):
  - a new test case generation procedure;
  - a logic to express linear properties of concurrent hierarchical computations;

This thesis work
An overview

- Verification of industrial controllers is the most critical and time-consuming phase in the development life-cycle.
- Techniques are sought to ease verification efforts and increase effectiveness.
- Extend previous research on Dynamic State Machines (DSTM):
  - a new test case generation procedure;
  - a logic to express linear properties of concurrent hierarchical computations;
  - a model checking procedure for hierarchical machines;

Outline

1. Systems verification.

Outline

1. Systems verification.
2. Previous research.

Outline

1. Systems verification.
2. Previous research.
3. Dynamic State Machines (DSTM).

Outline

1. Systems verification.
2. Previous research.
3. Dynamic State Machines (DSTM).
4. Test case generation for DSTM.

Outline

1. Systems verification.
2. Previous research.
3. Dynamic State Machines (DSTM).
4. Test case generation for DSTM.
5. A logic to predicate on concurrent hierarchical computations.

Outline

1. Systems verification.
2. Previous research.
3. Dynamic State Machines (DSTM).
4. Test case generation for DSTM.
5. A logic to predicate on concurrent hierarchical computations.
6. A model checking procedure for hierarchical machines.

Systems Verification

Systems verification
What it is all about

System verification

The process of checking that a system meets certain requirements derived from a given *specification*.

Systems verification
What it is all about

System verification

The process of checking that a system meets certain requirements derived from a given *specification*.

Why should we care?

Systems verification
What it is all about

System verification

The process of checking that a system meets certain requirements derived from a given *specification*.

Why should we care?

- Computer systems are ubiquitous and we depend more and more on them;

Systems verification
What it is all about

System verification

The process of checking that a system meets certain requirements derived from a given *specification*.

Why should we care?

- Computer systems are ubiquitous and we depend more and more on them;
- Malfunctions may cause financial losses.

Systems verification
What it is all about

System verification

The process of checking that a system meets certain requirements derived from a given *specification*.

Why should we care?

- Computer systems are ubiquitous and we depend more and more on them;
- Malfunctions may cause financial losses **or worse!**

Systems verification
When classic techniques fall short

Traditional verification techniques like testing and code inspection are effective at detecting errors.

Systems verification
When classic techniques fall short

Traditional verification techniques like testing and code inspection are effective at detecting errors, **but**:

Systems verification
When classic techniques fall short

Traditional verification techniques like testing and code inspection are effective at detecting errors, **but**:

- *hopelessly inadequate to prove their absence*!

Systems verification
When classic techniques fall short

Traditional verification techniques like testing and code inspection are effective at detecting errors, **but**:

- *hopelessly inadequate to prove their absence*!
- ineffective with complex concurrent systems;

Systems verification
When classic techniques fall short

Traditional verification techniques like testing and code inspection are effective at detecting errors, **but**:

- *hopelessly inadequate to prove their absence*!
- ineffective with complex concurrent systems;
- expensive.

Systems verification
The model-based approach

The model-based approach integrates formal methods in the verification process.

Systems verification
The model-based approach

The model-based approach integrates formal methods in the verification process.

- abstract a model (*formal specification*) of the system;

Systems verification
The model-based approach

The model-based approach integrates formal methods in the verification process.

- abstract a model (*formal specification*) of the system;
- apply model-based verification techniques:

Systems verification
The model-based approach

The model-based approach integrates formal methods in the verification process.

- abstract a model (*formal specification*) of the system;
- apply model-based verification techniques:
  - model-based testing (automatic test case generation);

Systems verification
The model-based approach

The model-based approach integrates formal methods in the verification process.

- abstract a model (*formal specification*) of the system;
- apply model-based verification techniques:
  - model-based testing (automatic test case generation);
  - model checking.

Model checking
The model-based approach

Model checking

Given a model and a property, check that the property holds in **every possible** model behaviour.

Model checking
The model-based approach

Model checking

Given a model and a property, check that the property holds in **every possible** model behaviour.

Model checking
The model-based approach

Model checking

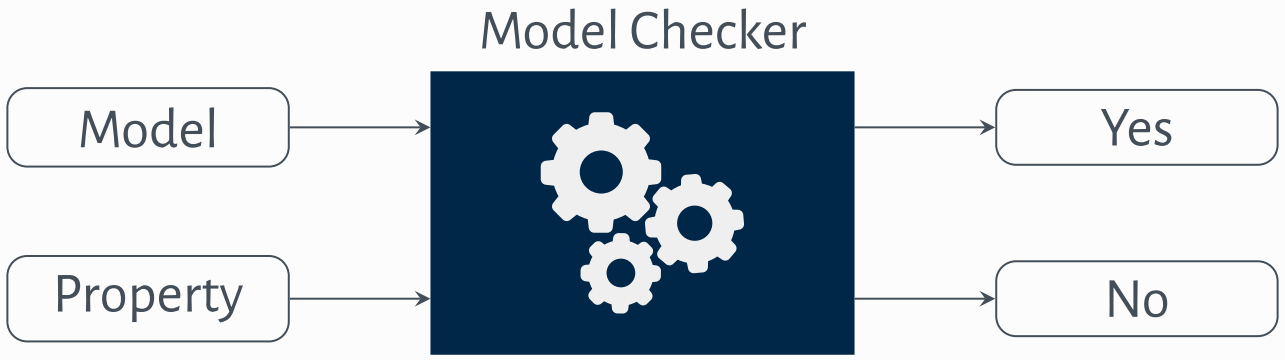
Given a model and a property, check that the property holds in **every possible** model behaviour.

- produces counter-examples;

Model checking
The model-based approach

Model checking

Given a model and a property, check that the property holds in **every possible** model behaviour.



- produces counter-examples;
- can be integrated earlier in the development process.

Previous research

The CRYSTAL project

CRitical sYSTems engineering AcceLeration

- driven by real-world industrial use cases from the automotive, aerospace, rail and health sector;
- produce or improve tool chains to reduce design/verification costs.

ARTEMIS

SEVENTH FRAMEWORK PROGRAMME

Previous research
within the CRYSTAL project

- introduced *Dynamic State Machines* (DSTM);

Previous research
within the CRYSTAL project

- introduced *Dynamic State Machines* (DSTM);
- devised an automatable test case generation procedure for DSTM;

Dynamic State Machines

Dynamic State Machines

A first example

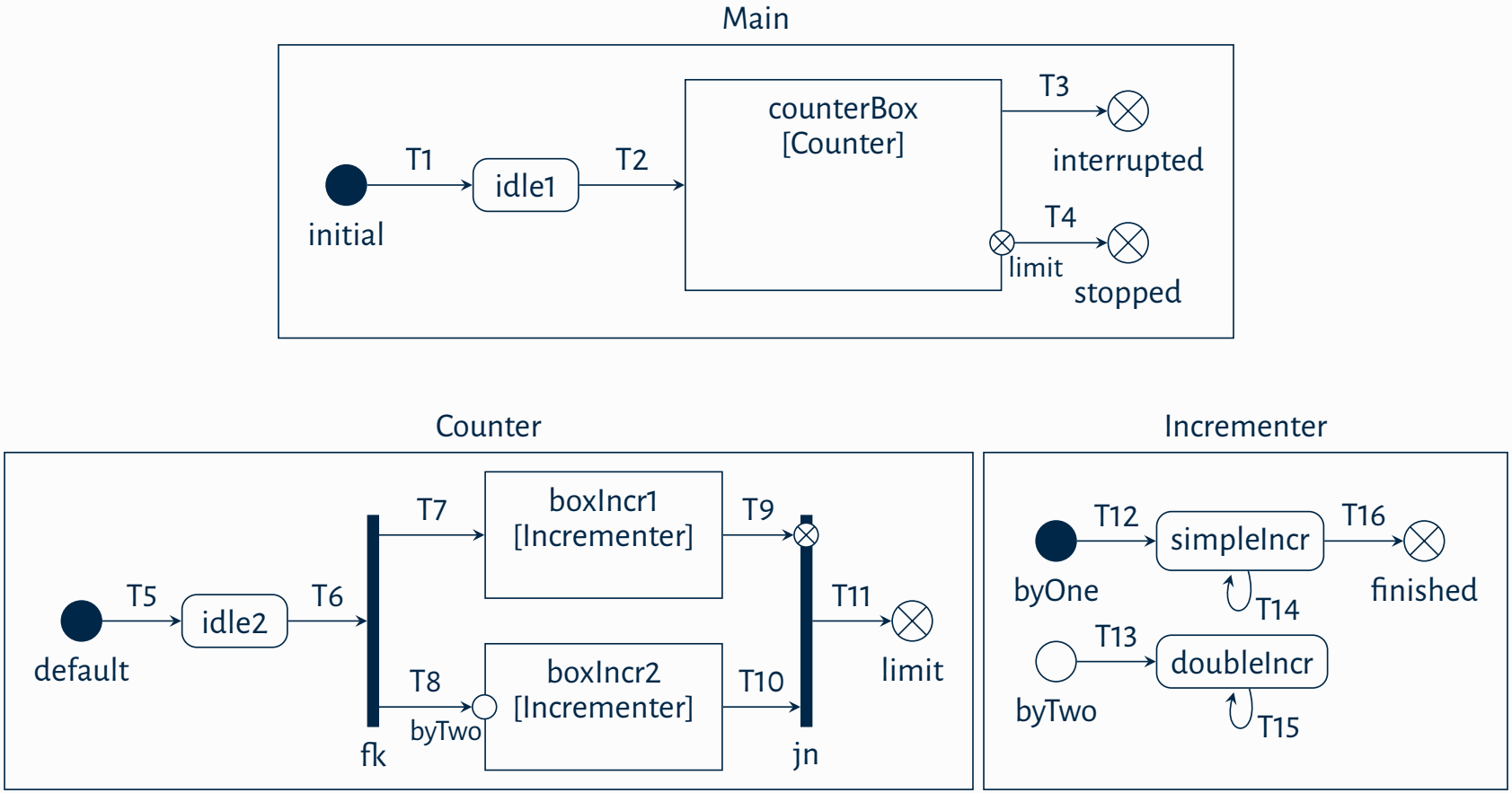
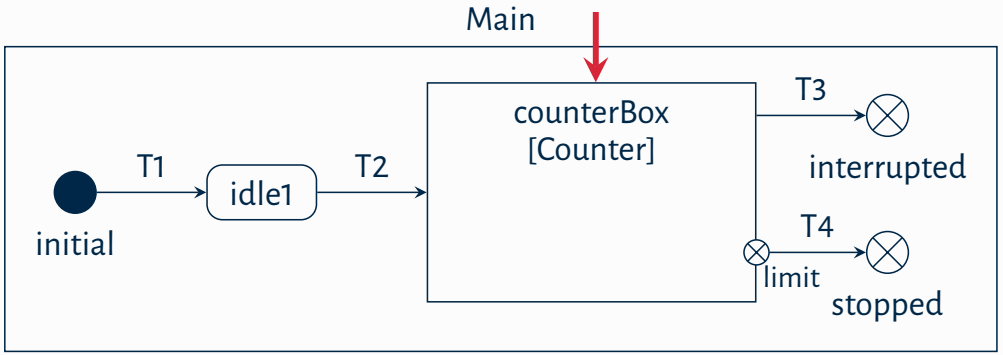
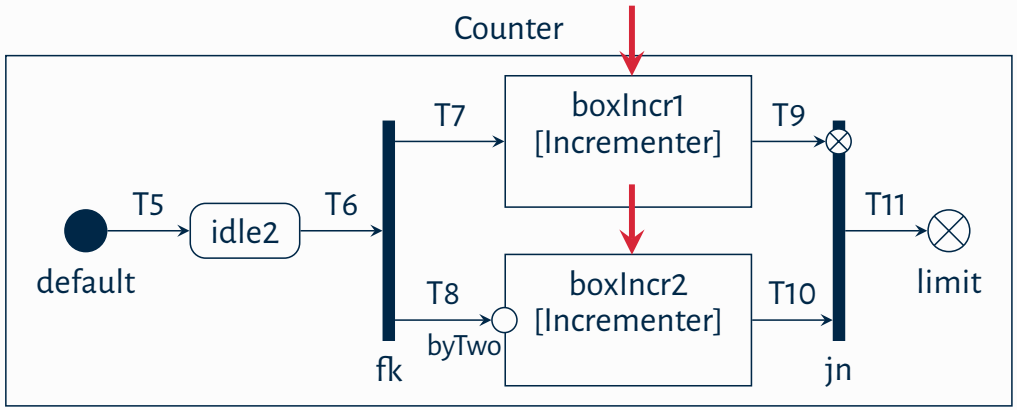


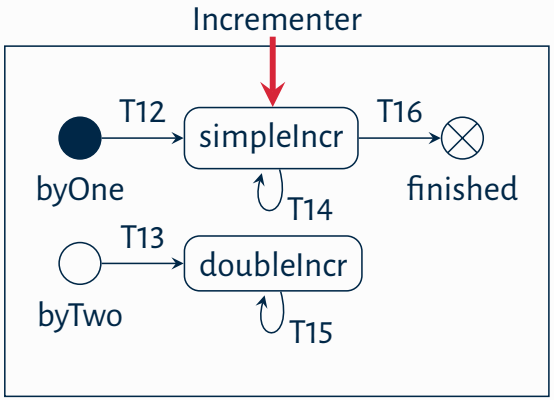
Figure: A simple DSTM specification

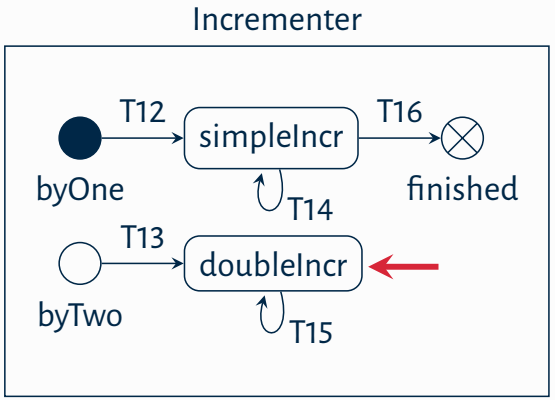
Dynamic State Machines

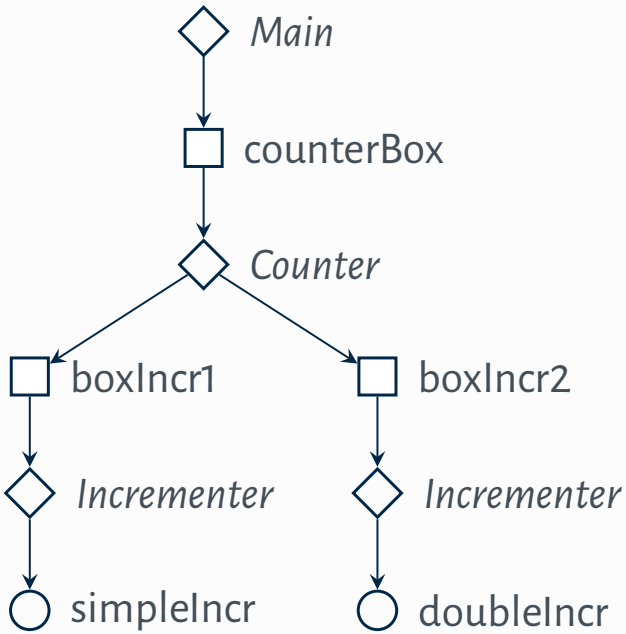
In action







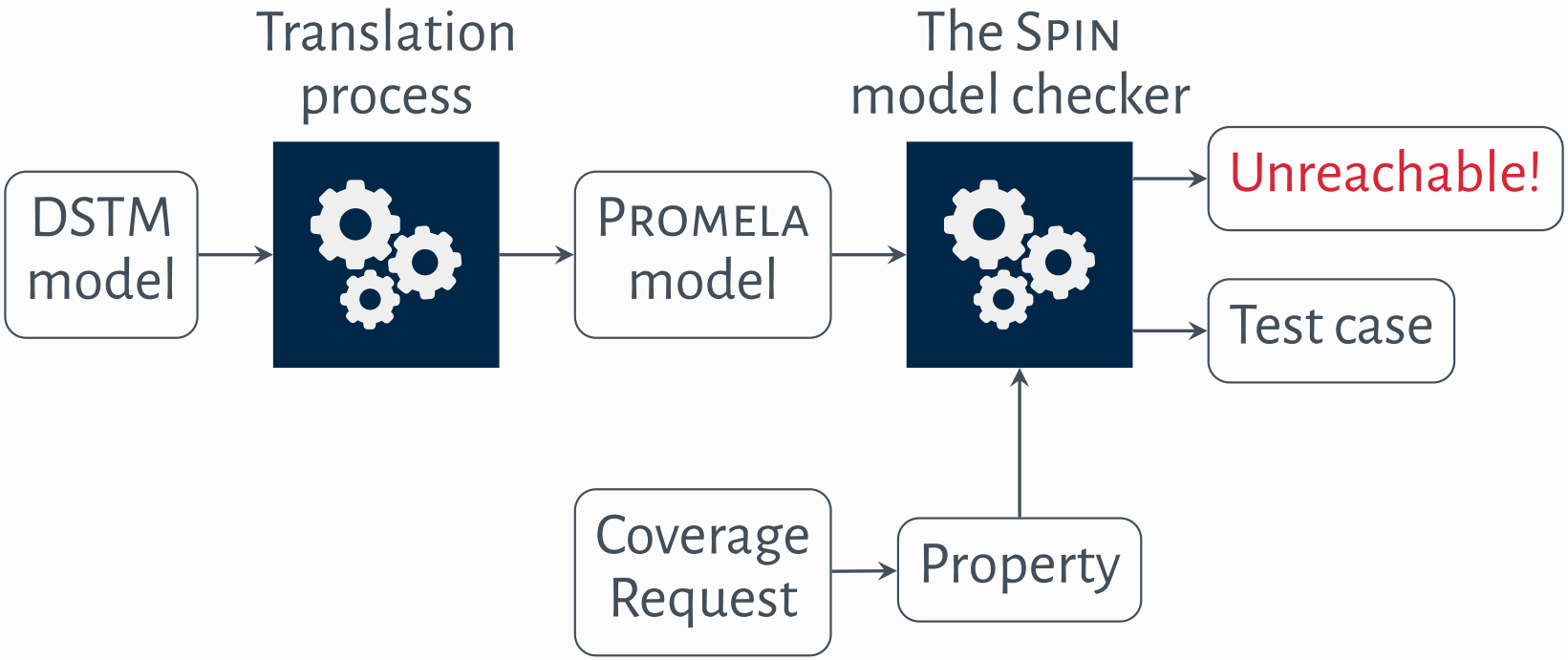




Model-based testing on DSTM models

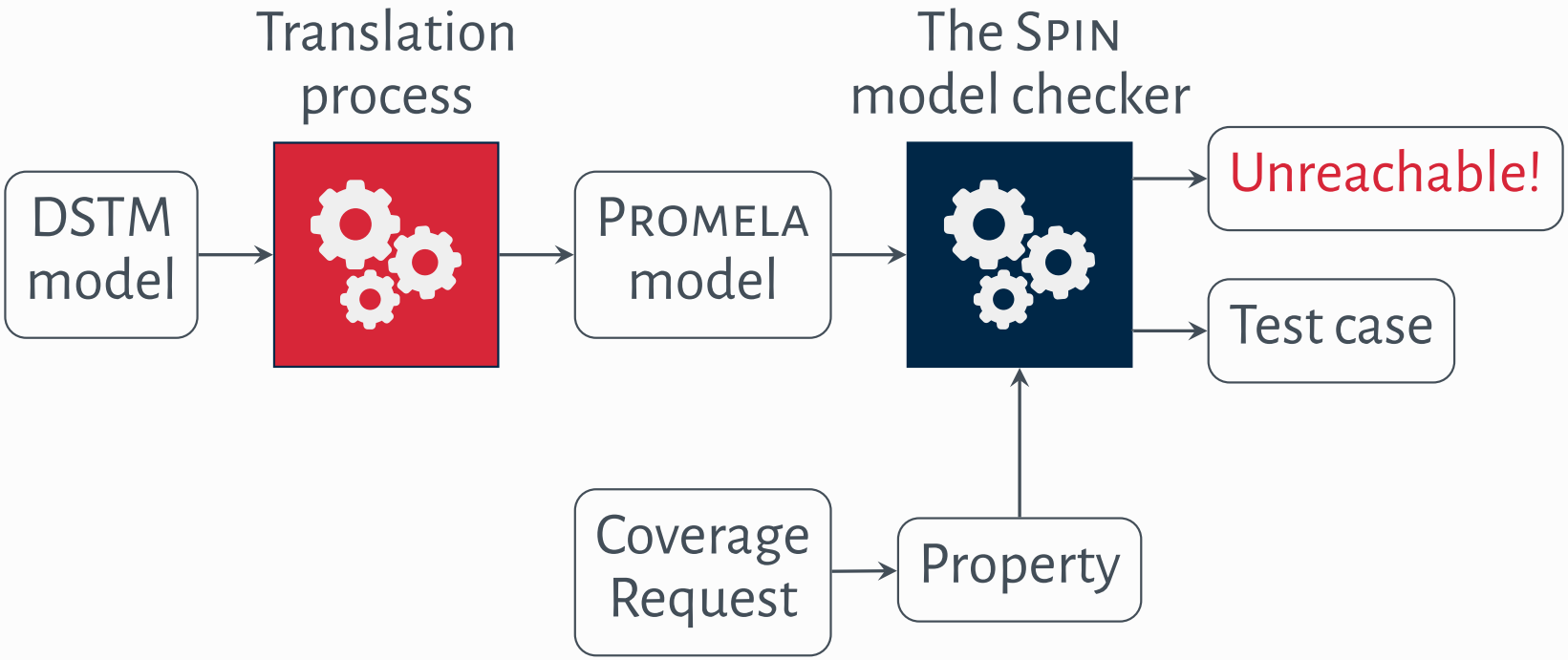
Model-based testing

Generating test-cases from DSTM models



Model-based testing

Generating test-cases from DSTM models



Hierarchical Linear-time Temporal Logic with Interrupts

Linear-time Temporal Logic (LTL)

Reasoning about linear time

$$\phi := p \in \mathcal{P} \mid \neg\phi \mid \phi \vee \phi$$

Linear-time Temporal Logic (LTL)

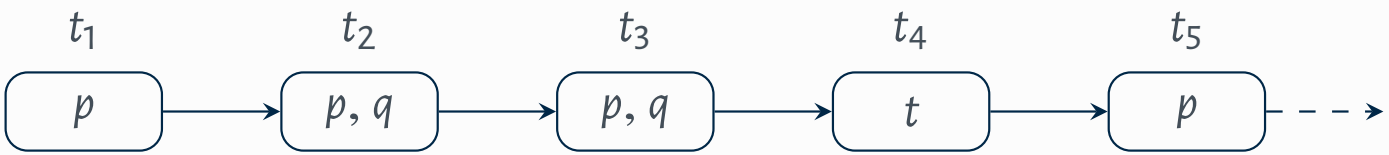
Reasoning about linear time

$$\phi := p \in \mathcal{P} \mid \neg\phi \mid \phi \vee \phi \mid \mathtt{X}\,\phi \mid \phi\,\mathtt{U}\,\phi$$

Linear-time Temporal Logic (LTL)

Reasoning about linear time

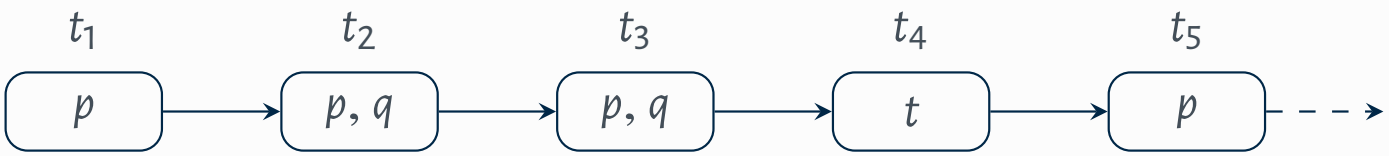
$$\phi := p \in \mathcal{P} \mid \neg\phi \mid \phi \vee \phi \mid \mathtt{X}\,\phi \mid \phi\,\mathtt{U}\,\phi$$



Linear-time Temporal Logic (LTL)

Reasoning about linear time

$$\phi := p \in \mathcal{P} \mid \neg\phi \mid \phi \vee \phi \mid \mathtt{X}\,\phi \mid \phi\,\mathtt{U}\,\phi$$

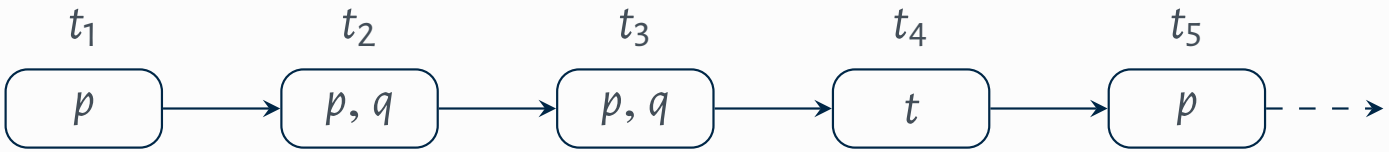


Some examples:

Linear-time Temporal Logic (LTL)

Reasoning about linear time

$$\phi := p \in \mathcal{P} \mid \neg\phi \mid \phi \vee \phi \mid \mathtt{X}\,\phi \mid \phi\,\mathtt{U}\,\phi$$



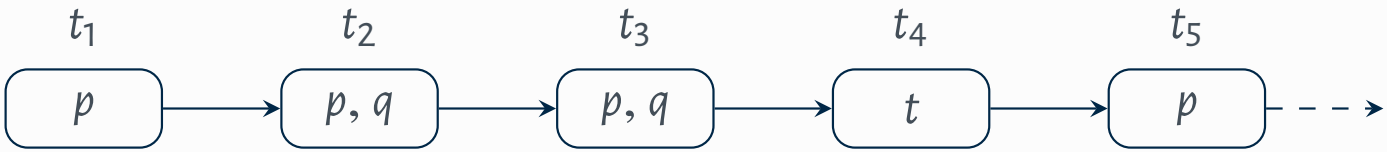
Some examples:

- $\mathtt{X}(p \wedge q)$ (Next)

Linear-time Temporal Logic (LTL)

Reasoning about linear time

$$\phi := p \in \mathcal{P} \mid \neg\phi \mid \phi \vee \phi \mid \mathtt{X}\,\phi \mid \phi\,\mathtt{U}\,\phi$$

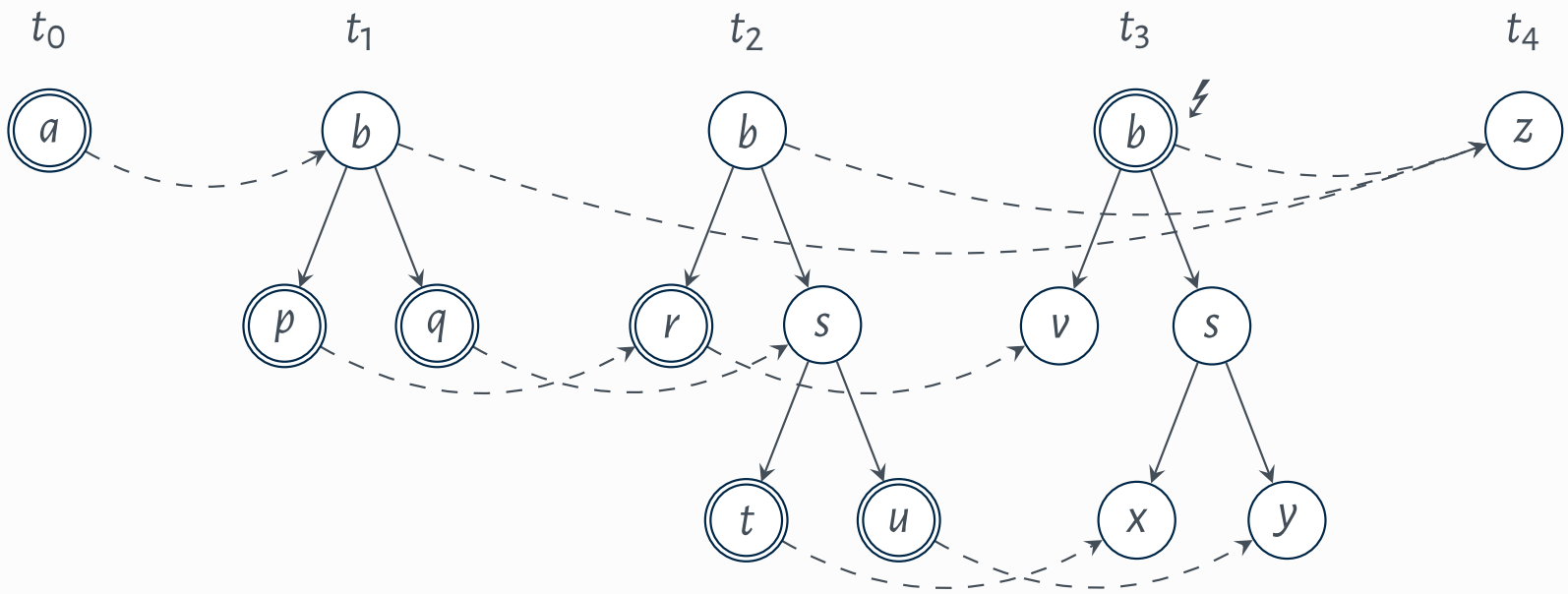


Some examples:

- $\mathtt{X}(p \wedge q)$ (Next)
- $p\,\mathtt{U}\,t$ (Until)

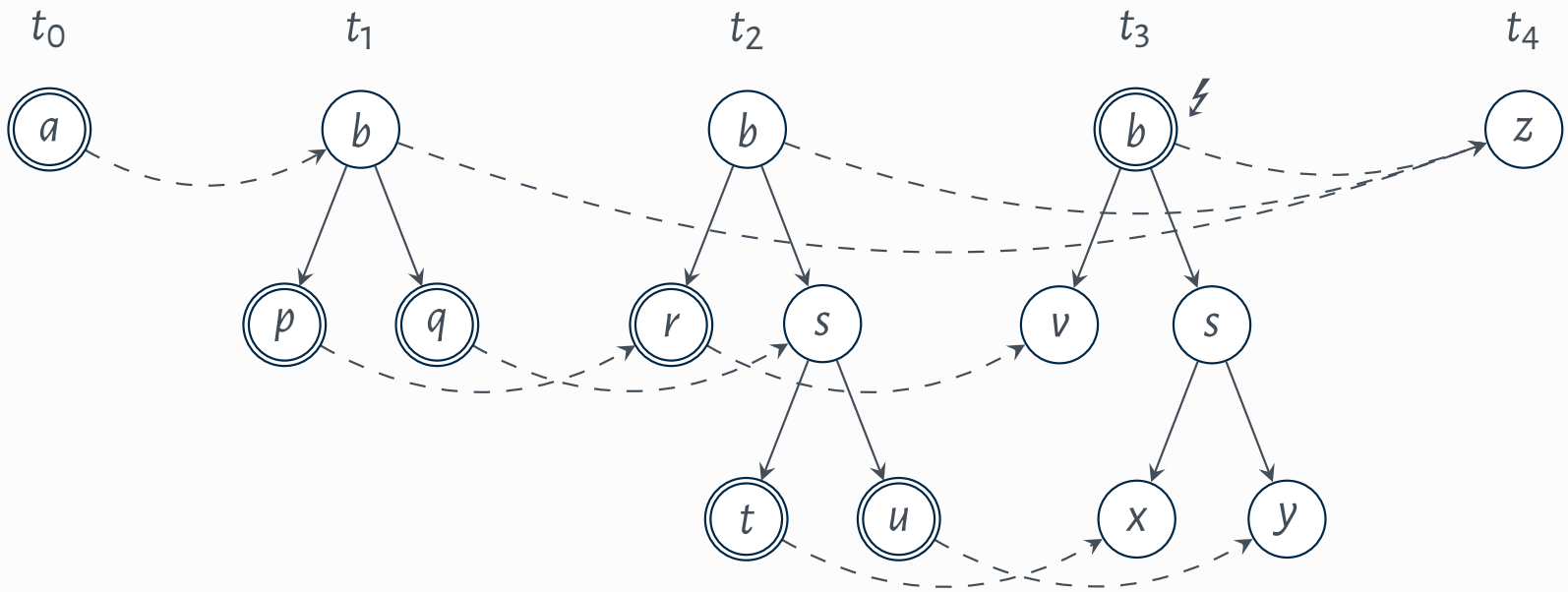
Hierarchical LTL with Interrupts

A novel extension of the well-known LTL



Hierarchical LTL with Interrupts

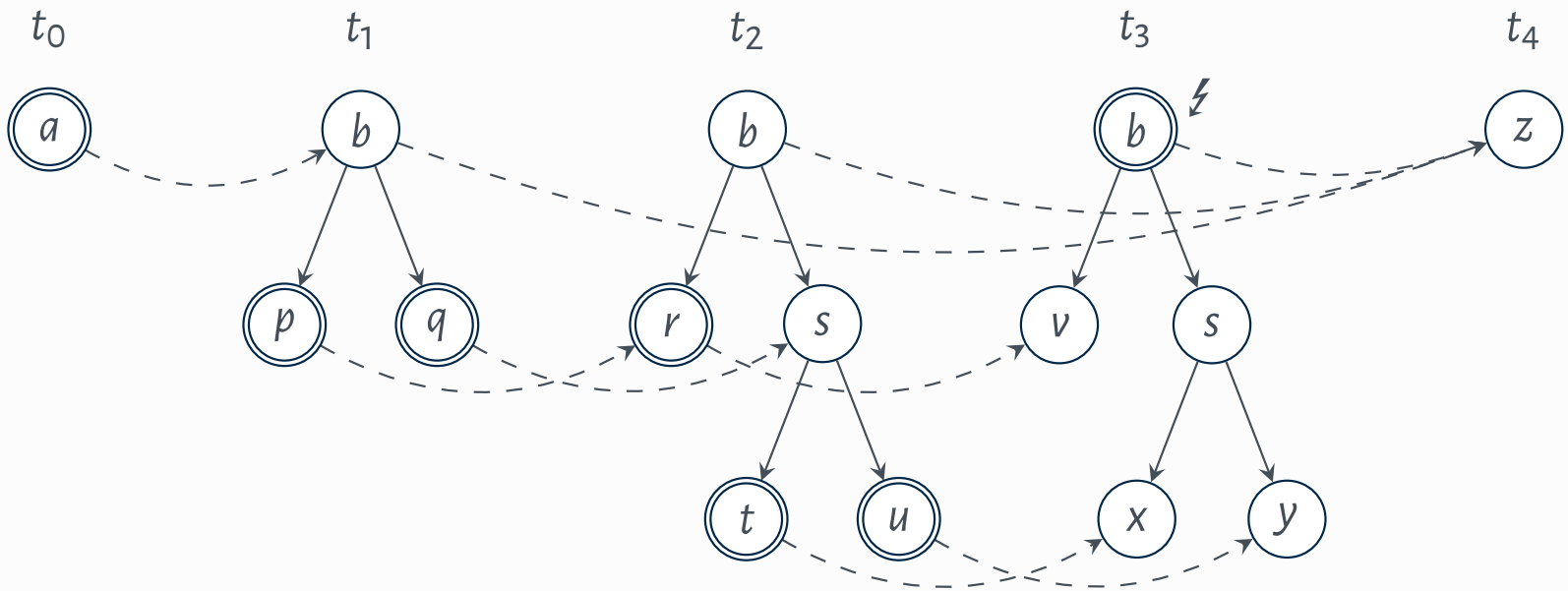
A novel extension of the well-known LTL



- LTL-like operators to reason about the evolution of a module;

Hierarchical LTL with Interrupts

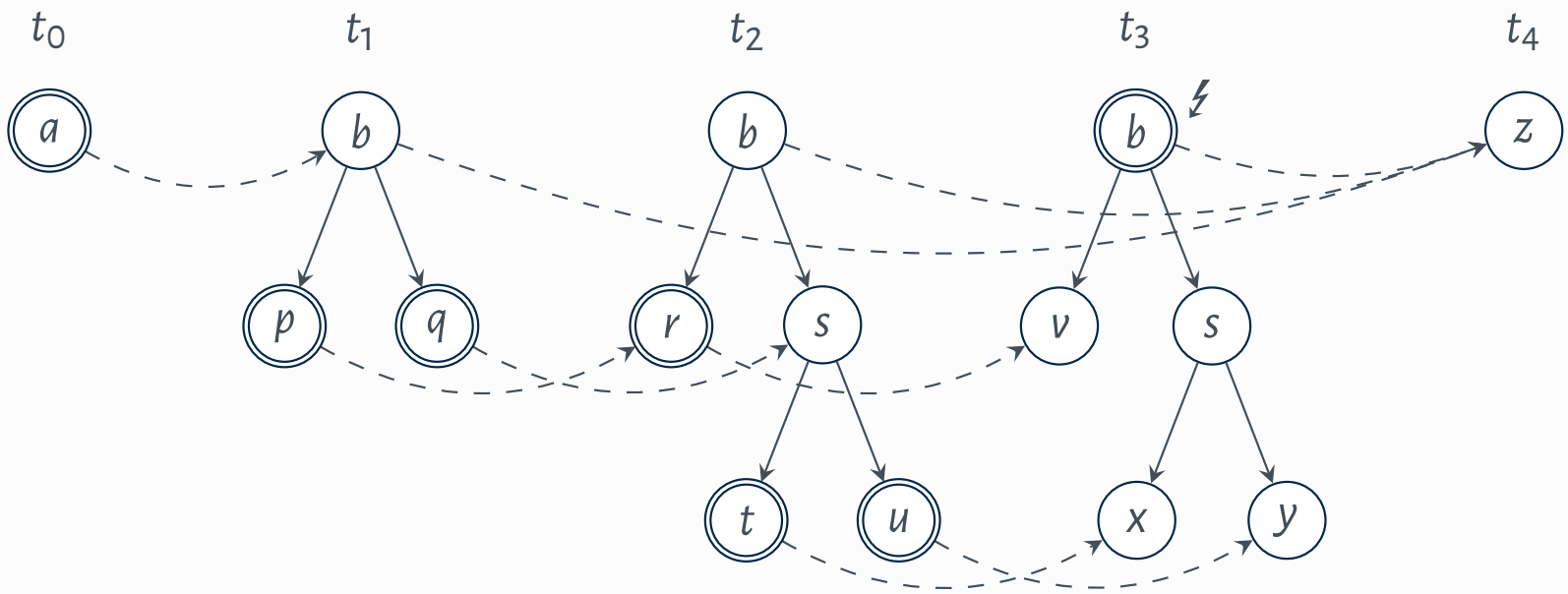
A novel extension of the well-known LTL



- LTL-like operators to reason about the evolution of a module;
- new operators to contextualize formulae in the hierarchy of modules.

Hierarchical LTL with Interrupts

A novel extension of the well-known LTL

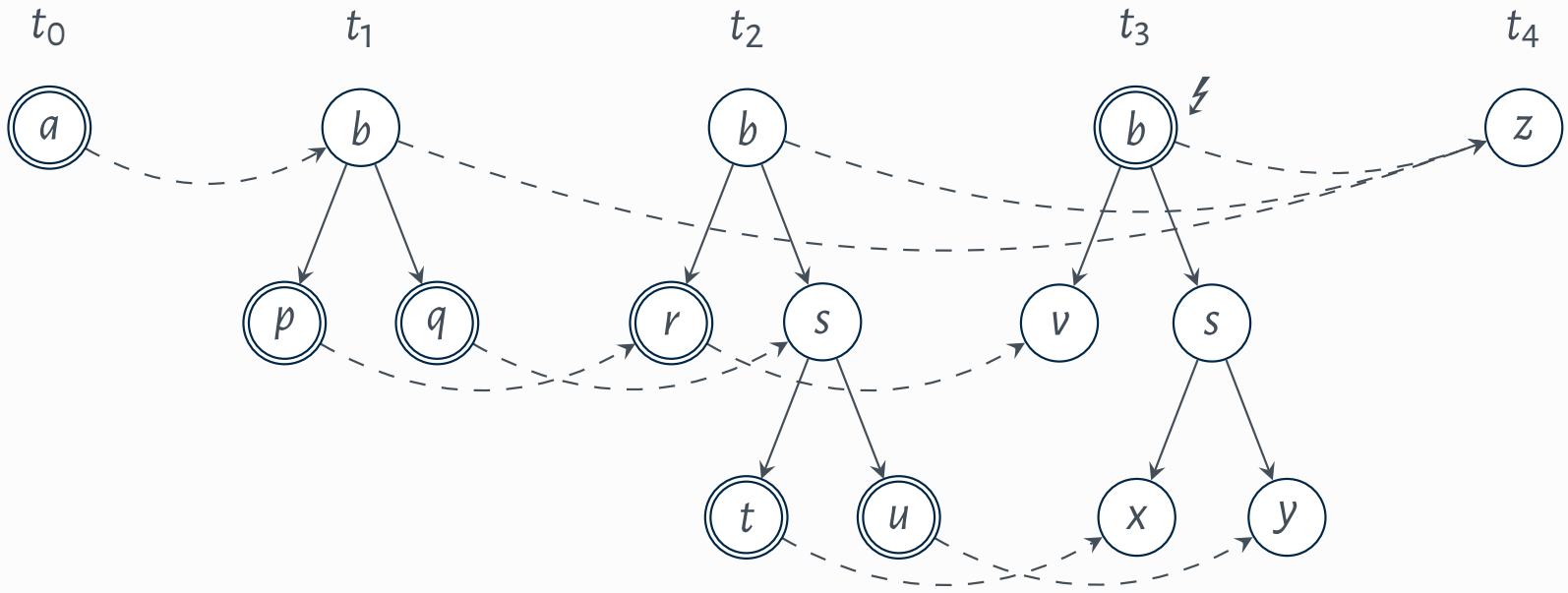


New operators:

- $\mathrm{X}_\lightning\, \phi$ (interrupting next);

Hierarchical LTL with Interrupts

A novel extension of the well-known LTL

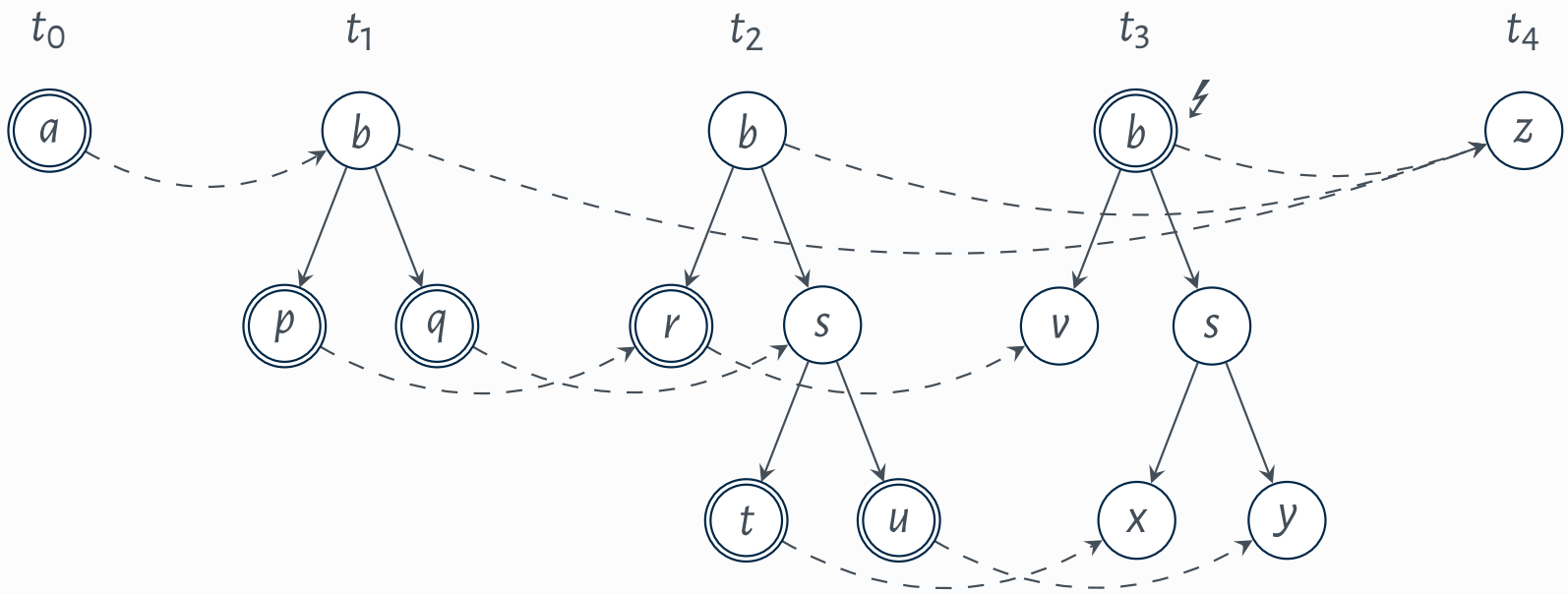


New operators:

- $\mathrm{X}_{\lightning}\,\phi$ (interrupting next);
- $\downarrow_i(\phi)$, $\leftarrow(\phi)$, $\rightarrow(\phi)$ (contextualization operators).

Hierarchical LTL with Interrupts

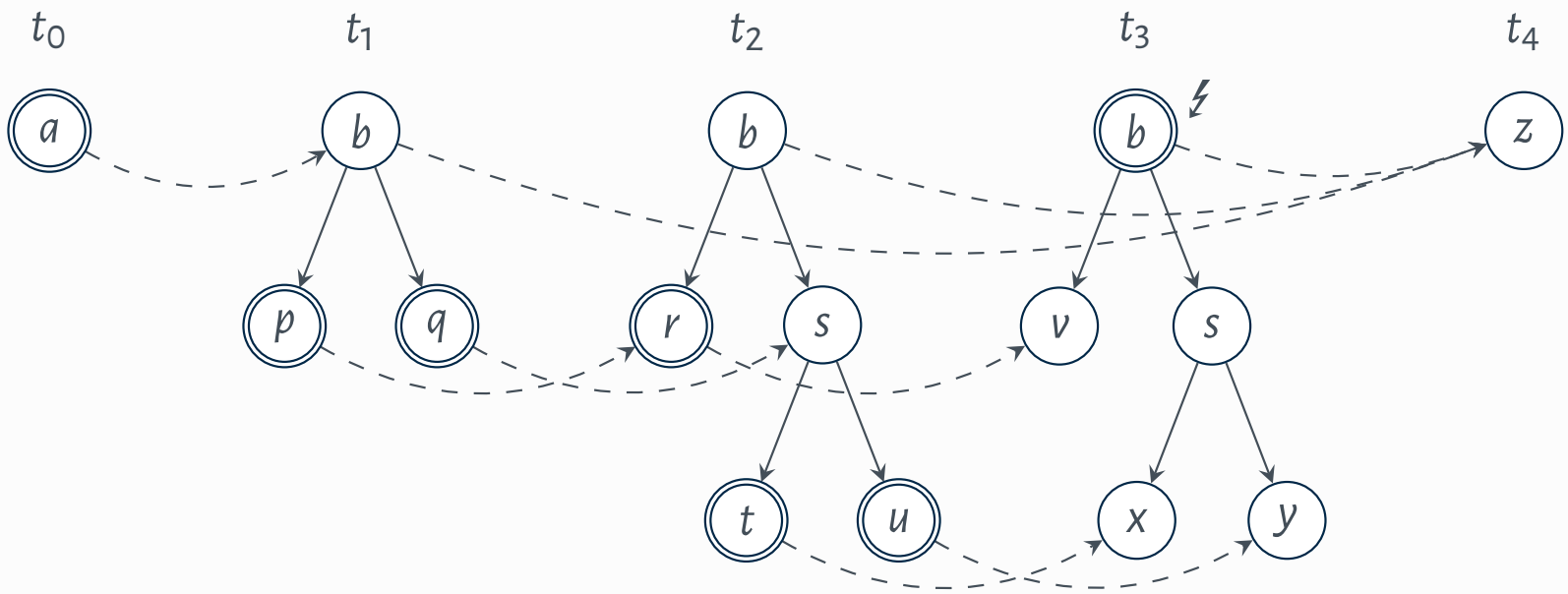
A novel extension of the well-known LTL



Some examples:

Hierarchical LTL with Interrupts

A novel extension of the well-known LTL

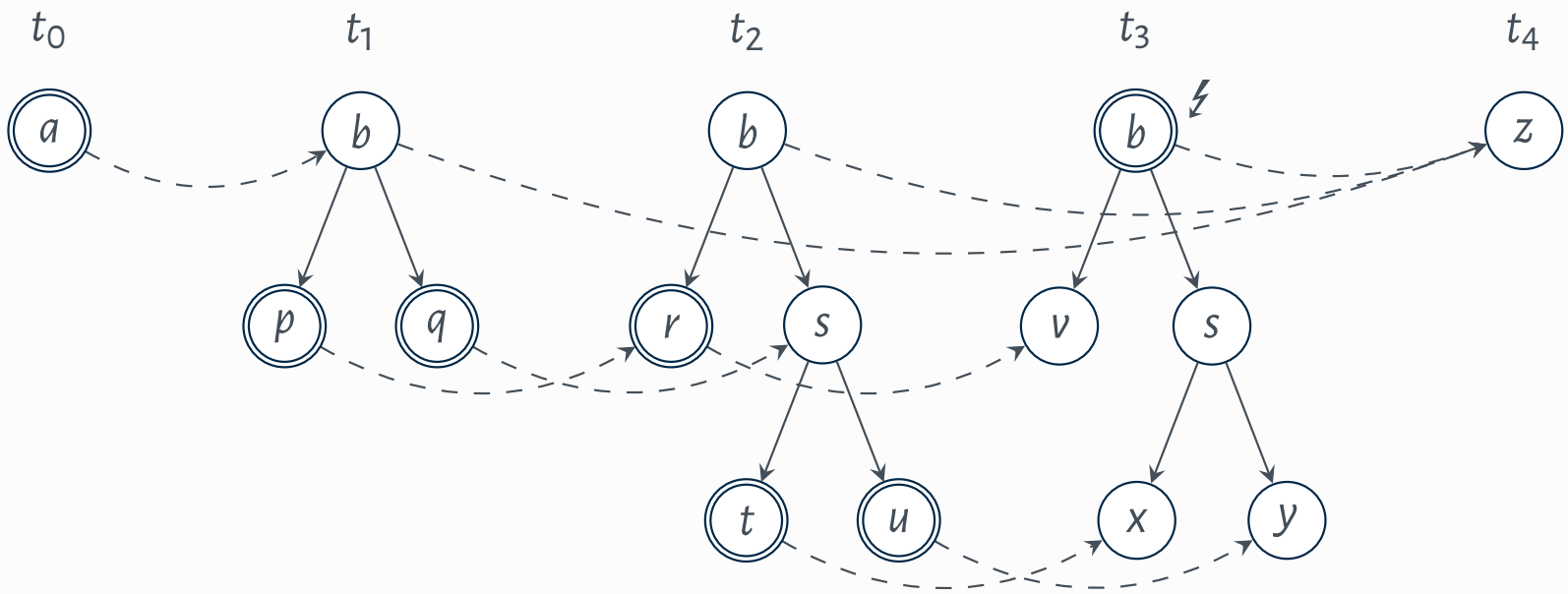


Some examples:

- $\mathrm{X}_{\lightning}(z)$;

Hierarchical LTL with Interrupts

A novel extension of the well-known LTL



Some examples:

- $\mathrm{X}_{\lightning}(z)$;
- $\mathrm{X}\left(\downarrow_1(p \wedge \rightarrow(q))\right)$;

Communicating Hierarchical Automata

Simplifying DSTM

Communicating Hierarchical Automata with Interrupts (CHA$^{\lightning}$)

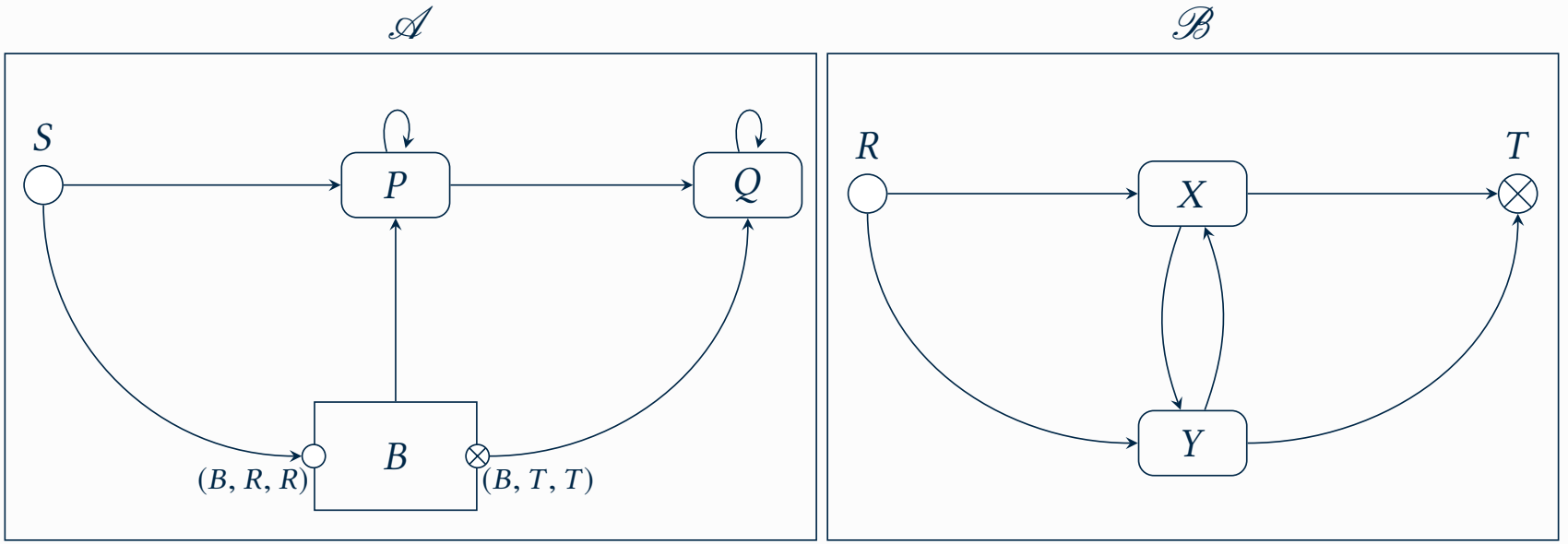


Figure: A simple CHA$^{\lightning}$

Simplifying DSTM

Communicating Hierarchical Automata with Interrupts (CHA$^{\lightning}$)

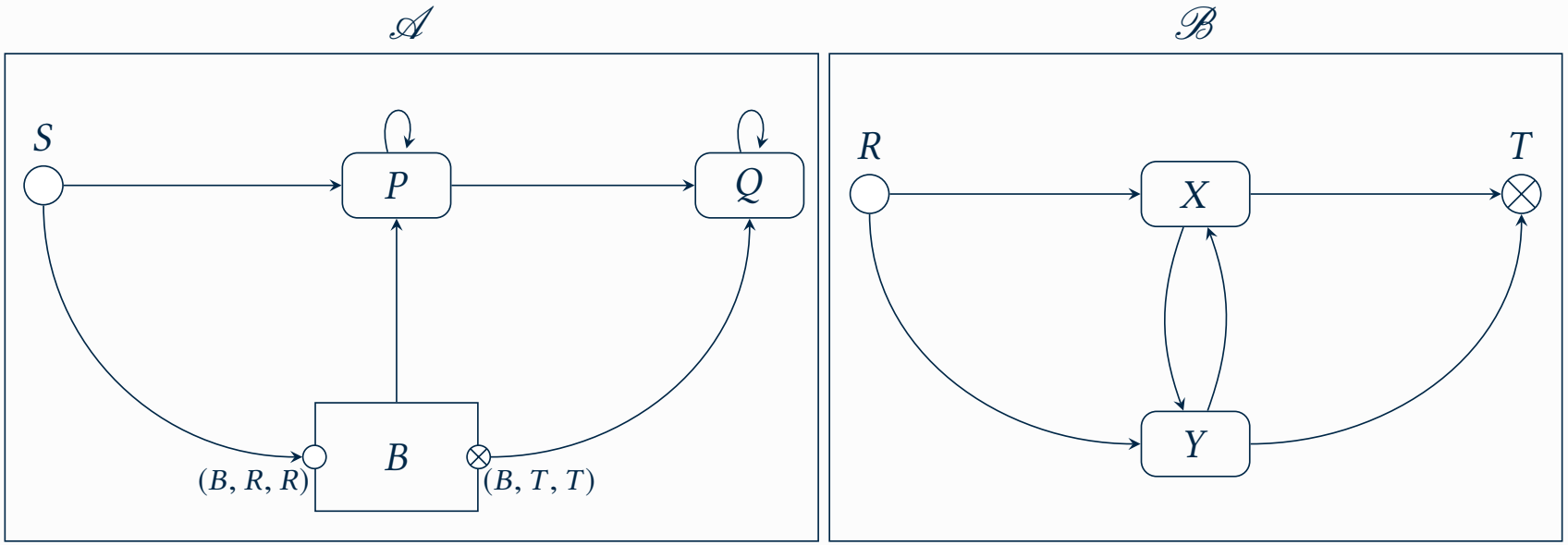


Figure: A simple CHA$^{\lightning}$

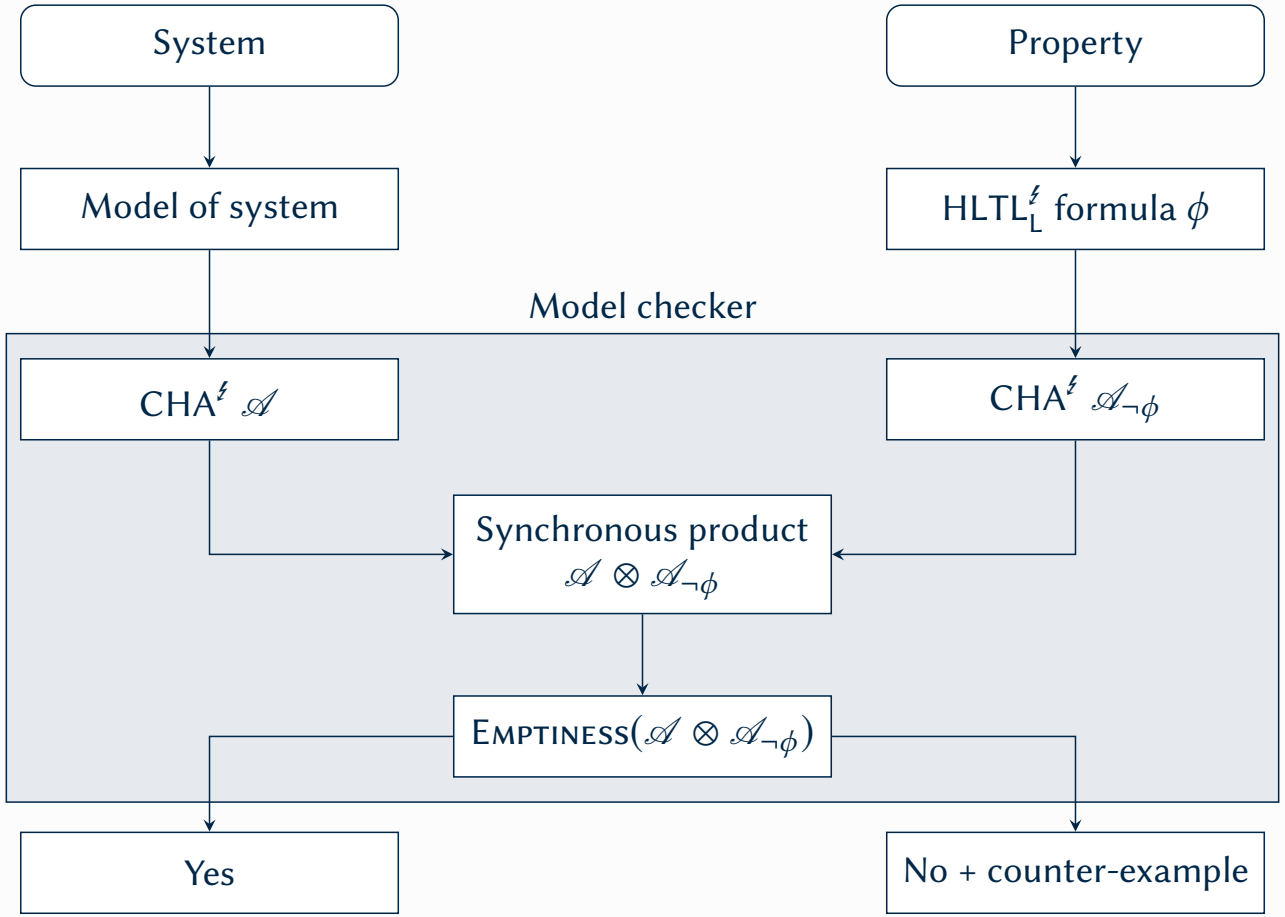
Differences with DSTM:

- no recursion;
- no forks/joins;

Model checking

The model checking procedure

Overview



Conclusions

Conclusions
Achieved goals and further research

Achieved goals:

Conclusions
Achieved goals and further research

Achieved goals:

- detected and addressed critical flaws in the DSTM test case generation procedure;

Conclusions
Achieved goals and further research

Achieved goals:

- detected and addressed critical flaws in the DSTM test case generation procedure;
- proposed HLTL$^{\lightning}$ to expressively predicate on concurrent interrupting hierarchical computations;

Conclusions

Achieved goals and further research

Achieved goals:

- detected and addressed critical flaws in the DSTM test case generation procedure;
- proposed HLTL$^{\lightning}$ to expressively predicate on concurrent interrupting hierarchical computations;
- devised PSPACE decision procedure model checking (CHA$^{\lightning}$ emptiness, synthesis, satisfiability, ...).

Conclusions

Achieved goals and further research

Achieved goals:

- detected and addressed critical flaws in the DSTM test case generation procedure;
- proposed HLTL$^{\lightning}$ to expressively predicate on concurrent interrupting hierarchical computations;
- devised PSPACE decision procedure model checking (CHA$^{\lightning}$ emptiness, synthesis, satisfiability, ...).

Further work:

Conclusions
Achieved goals and further research

Achieved goals:

- detected and addressed critical flaws in the DSTM test case generation procedure;
- proposed HLTL$^{\lightning}$ to expressively predicate on concurrent interrupting hierarchical computations;
- devised PSPACE decision procedure model checking (CHA$^{\lightning}$ emptiness, synthesis, satisfiability, ...).

Further work:

- extend results to DSTM;

Conclusions
Achieved goals and further research

Achieved goals:

- detected and addressed critical flaws in the DSTM test case generation procedure;
- proposed HLTL$^{\lightning}$ to expressively predicate on concurrent interrupting hierarchical computations;
- devised PSPACE decision procedure model checking (CHA$^{\lightning}$ emptiness, synthesis, satisfiability, ...).

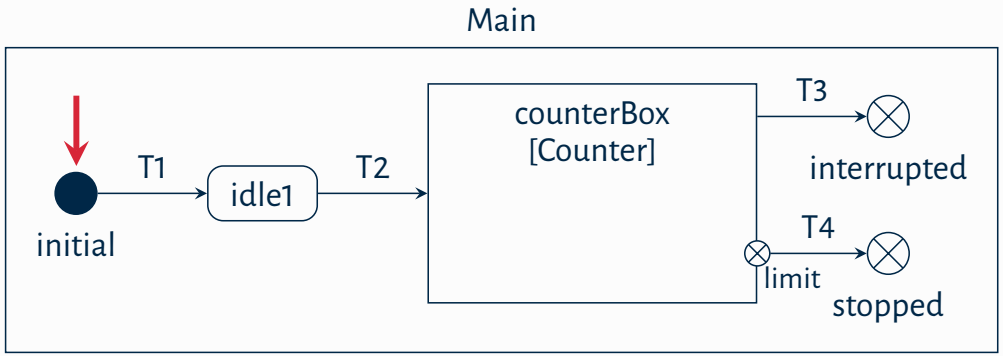
Further work:

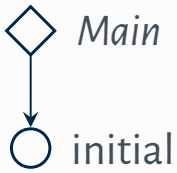
- extend results to DSTM;
- implement model checking in the existing DSTM tool chain.

Thank you for your time!

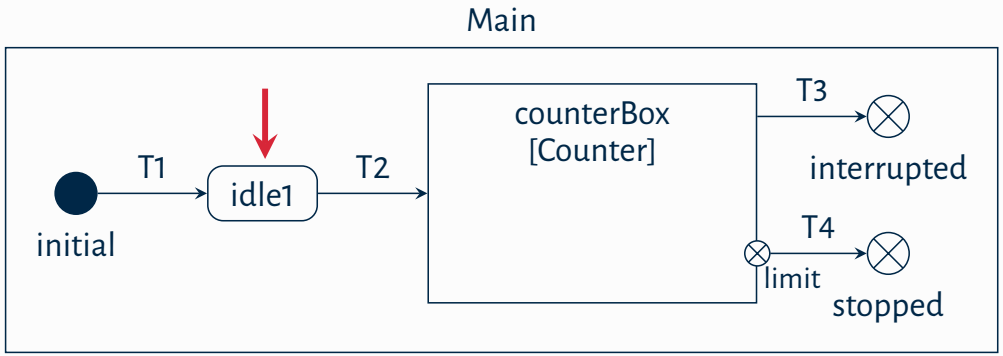
Backup slides

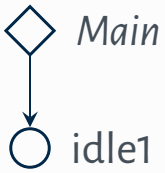
Dynamic State Machines
In action





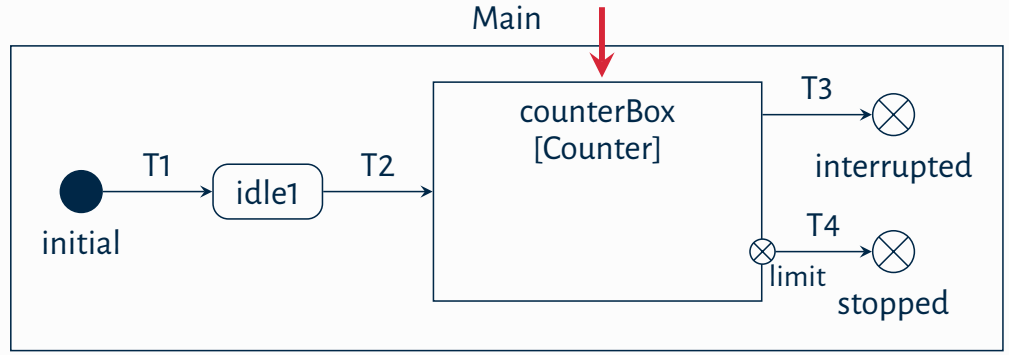
Dynamic State Machines
In action

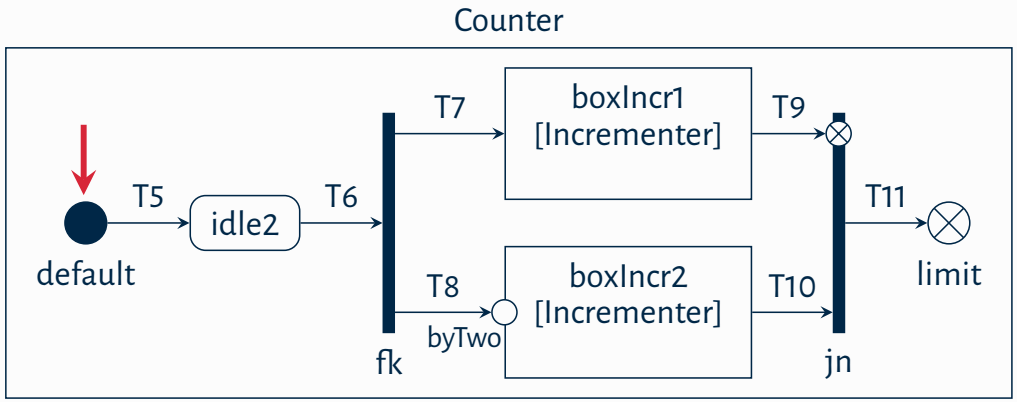


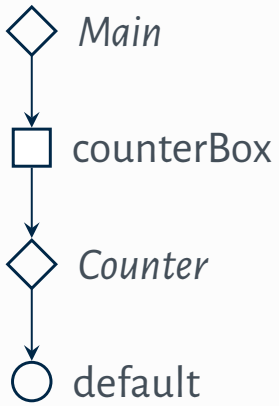


Dynamic State Machines

In action

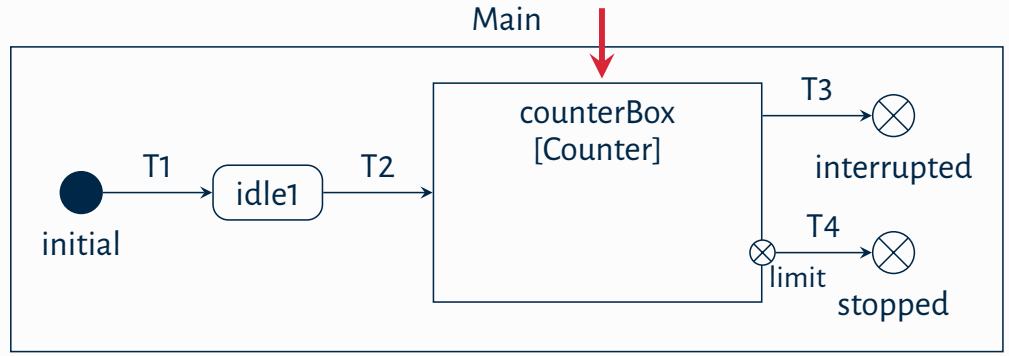


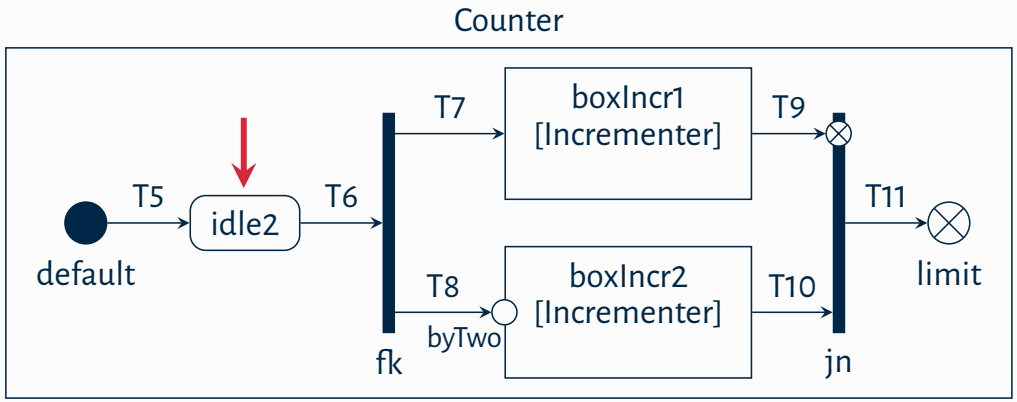


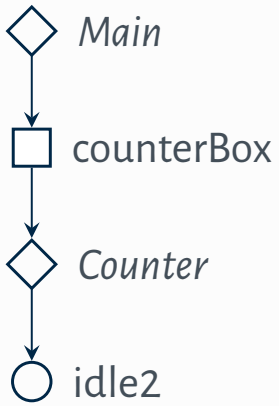


Dynamic State Machines

In action

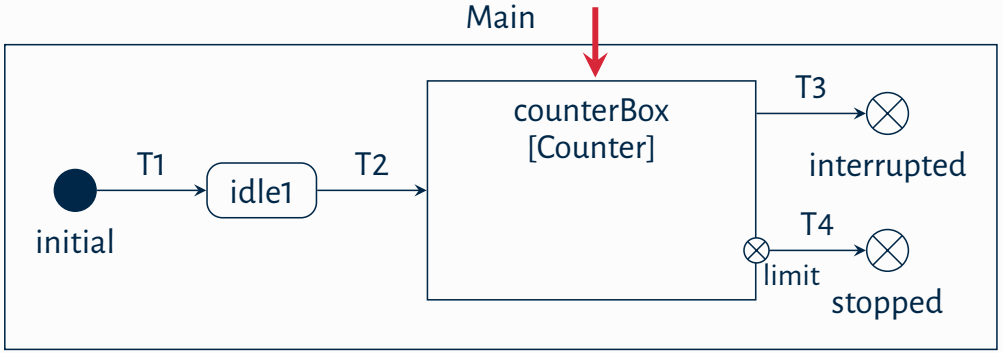


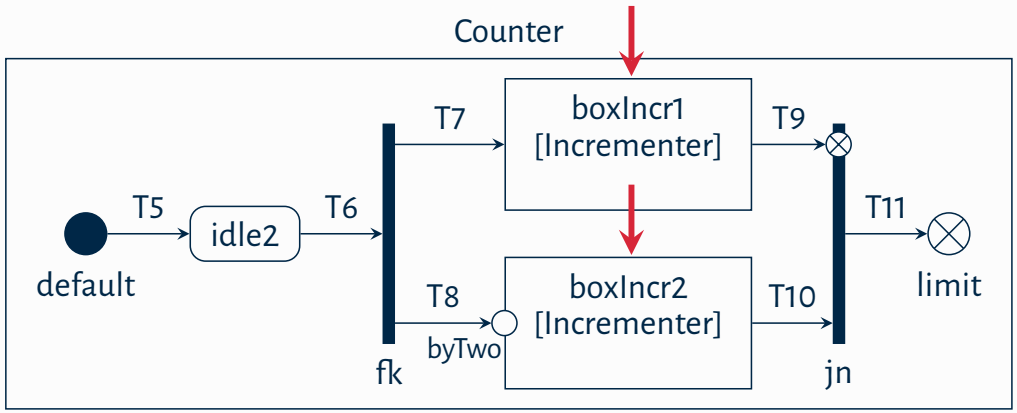


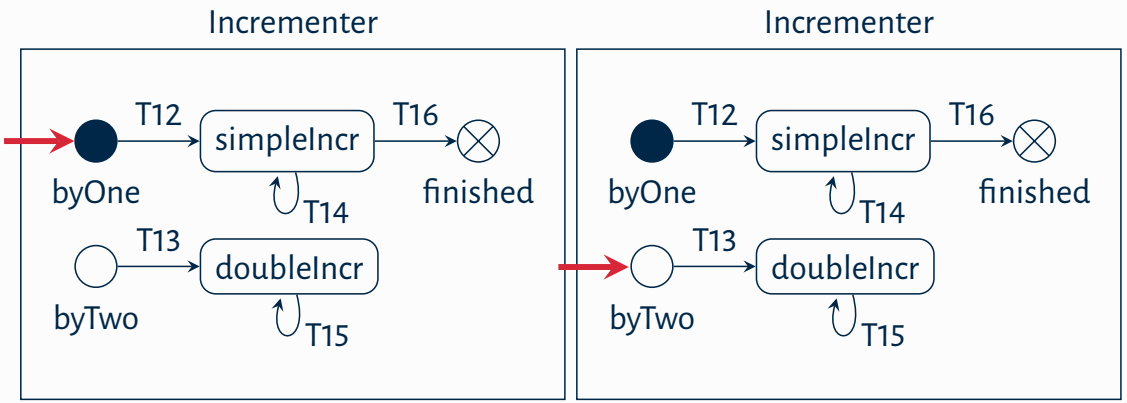


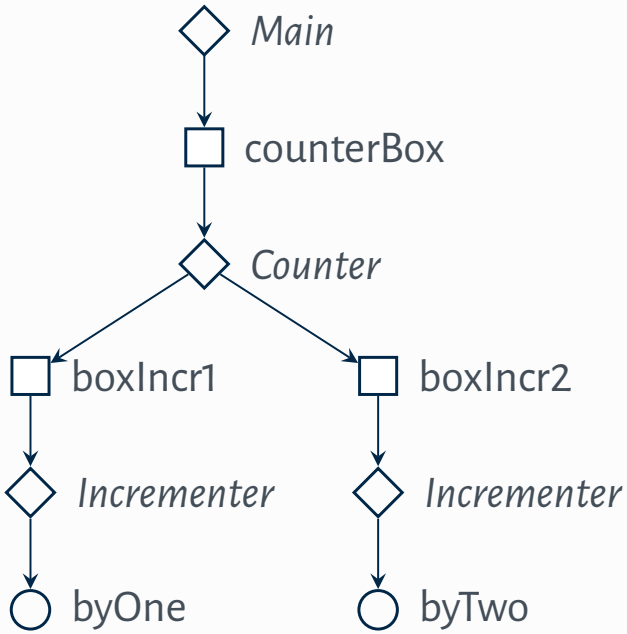
Dynamic State Machines

In action



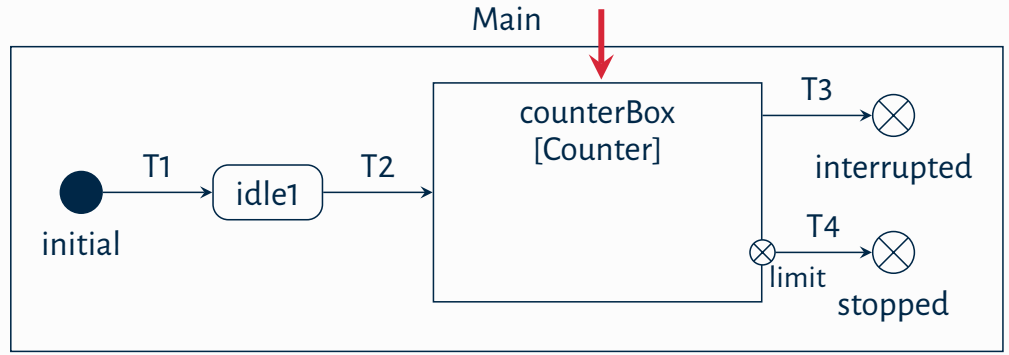


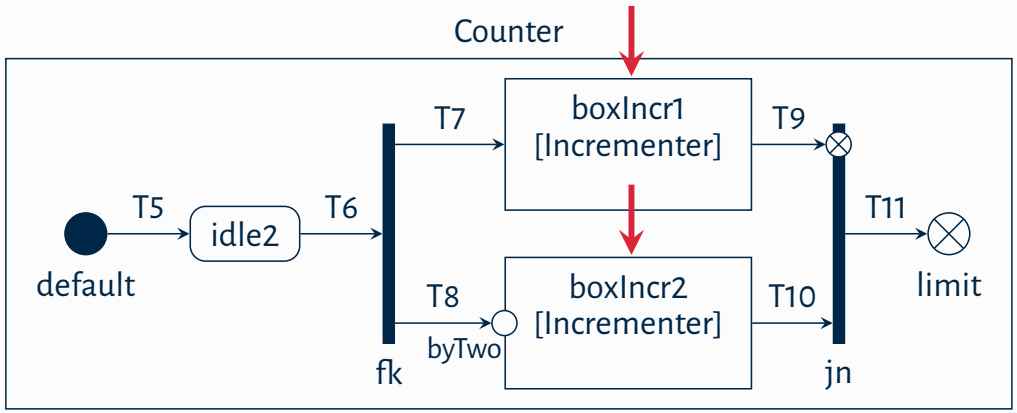


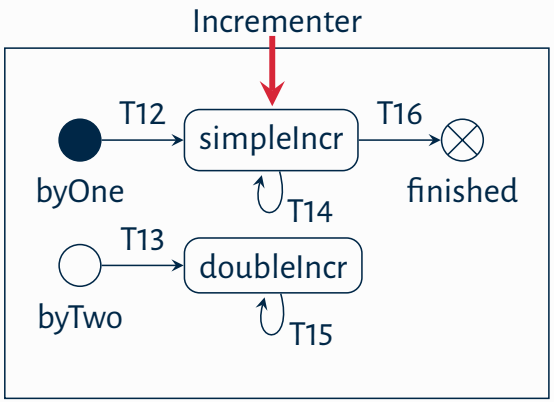


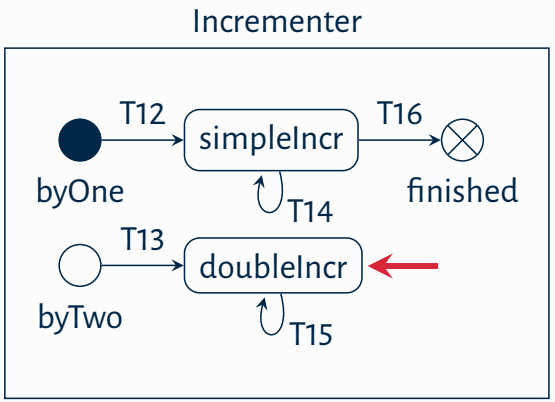
Dynamic State Machines

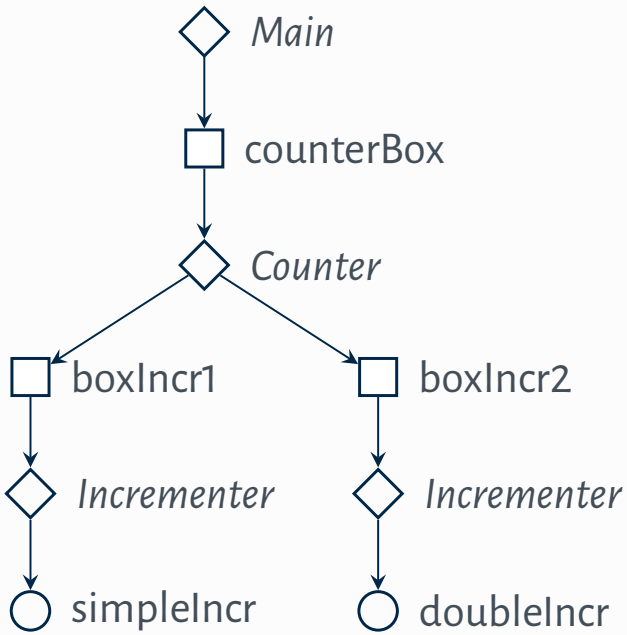
In action





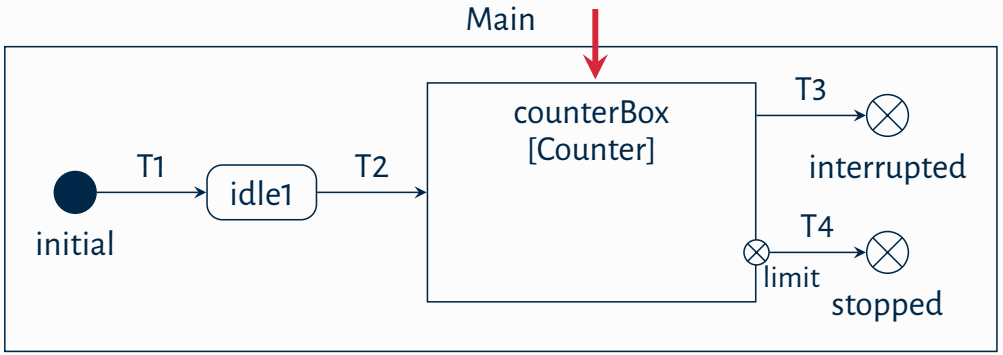


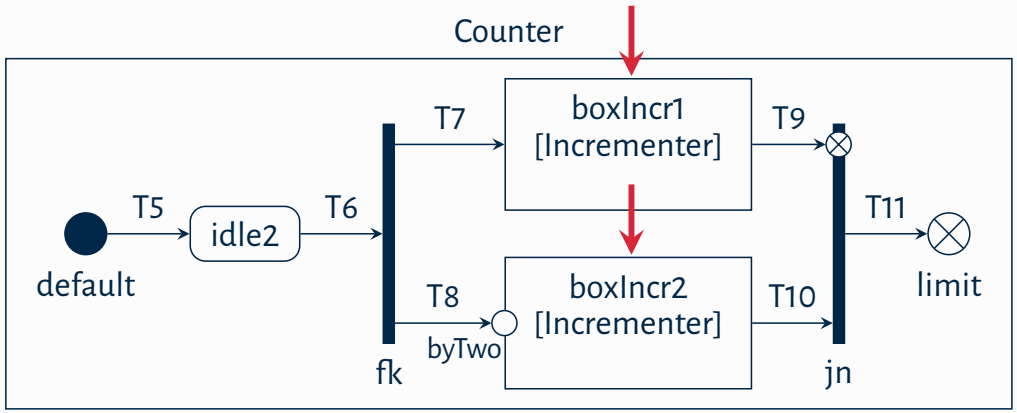


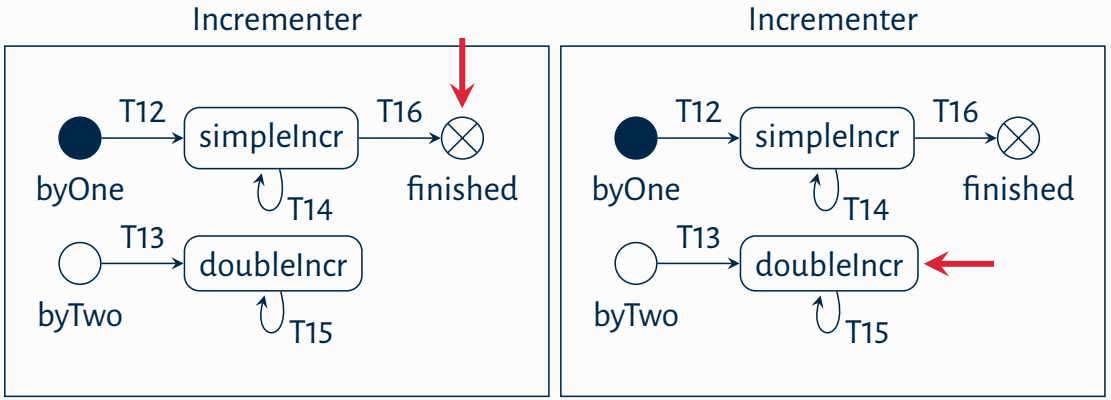


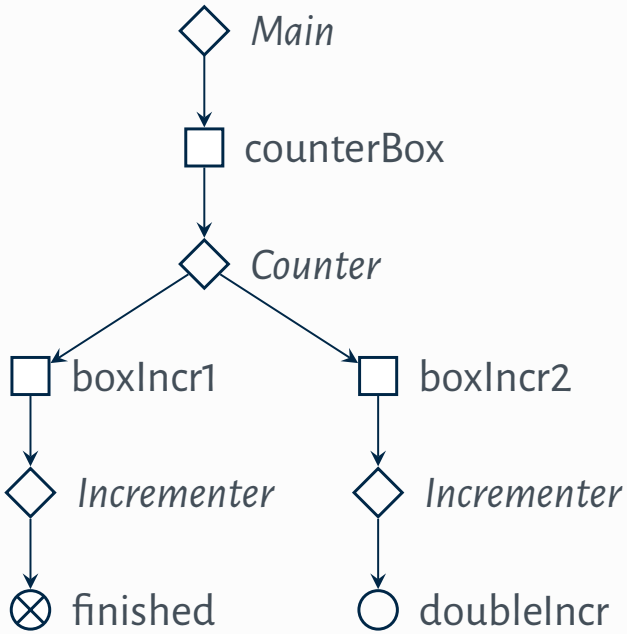
Dynamic State Machines

In action



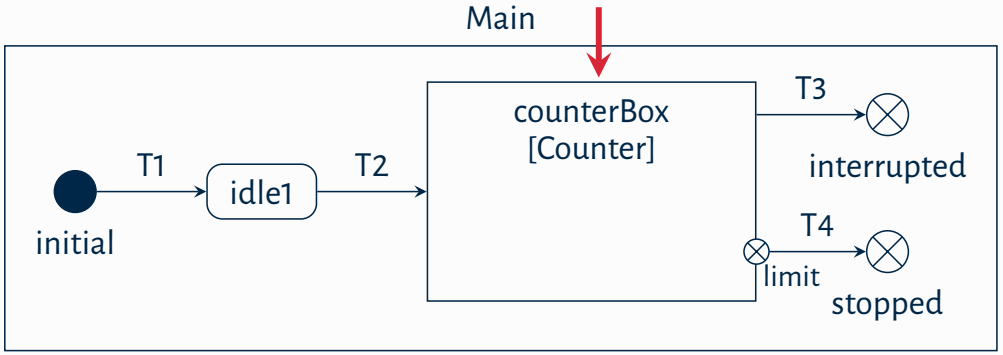


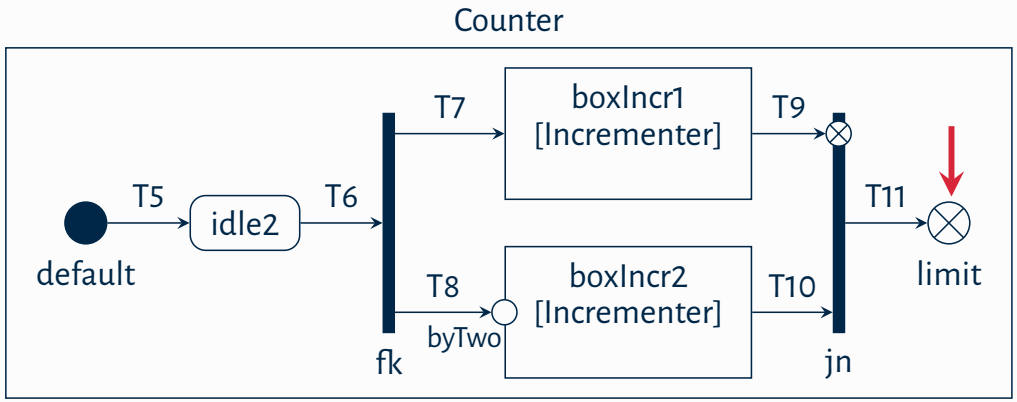


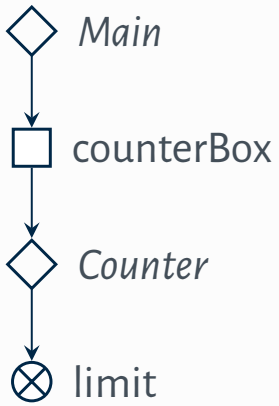


Dynamic State Machines

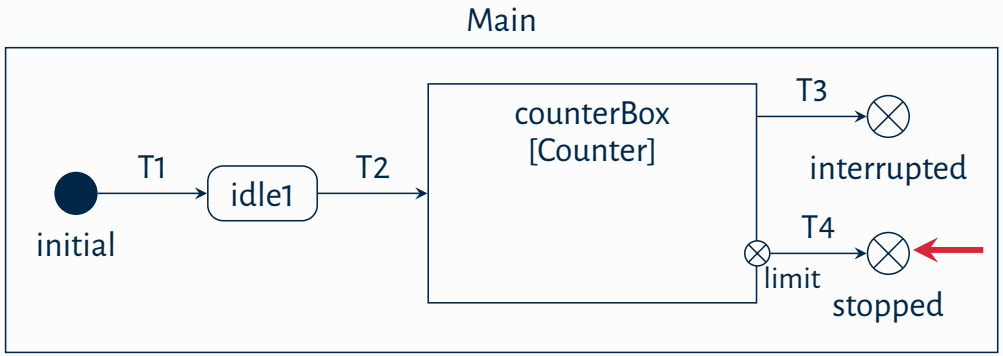
In action

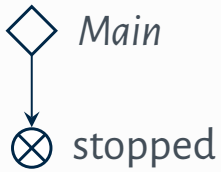






Dynamic State Machines
In action





Dynamic State Machines

A dynamic instantiation example

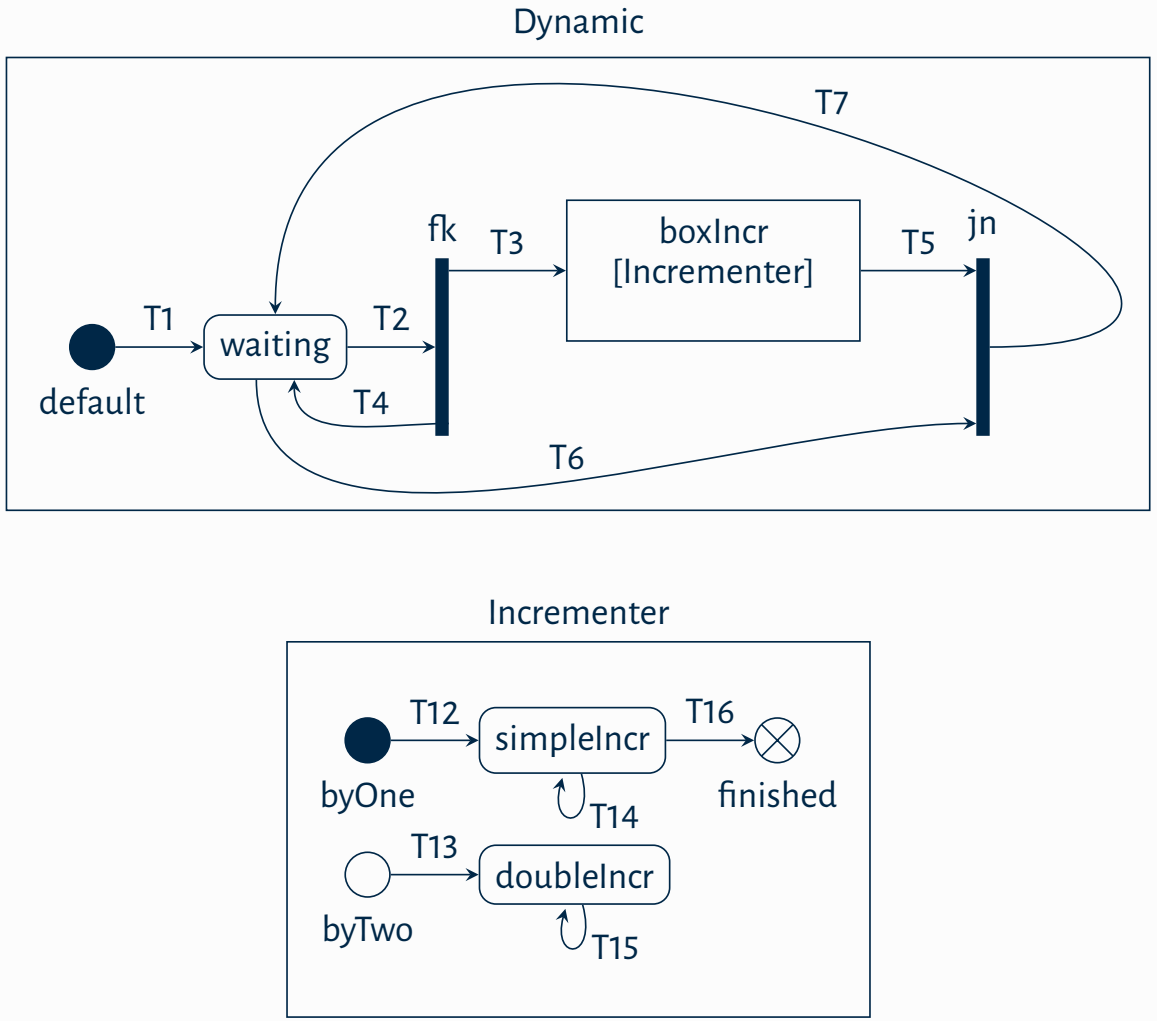


Figure: A dynamic DSTM specification

Deciding Emptiness for CSA

The Emptiness procedure

Emptiness problem

Given a CHA$^{\lightning}$, is there any accepting computation?

Main intuition:

- bottom-up summarization;

Deciding Satisfiability for HLTL$^{\lightning}$ formulae

The Satisfiability procedure

Satisfiability problem

Given a HLTL$^{\lightning}$ formula ϕ, is there any hierarchical computation satisfying ϕ?

Main intuitions:

- build a CHA$^{\lightning}$ accepting hierarchical words satisfying the formula;
- non-trivial extension of the Vardi-Wolper approach for LTL;
- check for the emptiness of the automaton;

The model checking procedure

Overview