Trends in Concurrency Theory
The Tester's Perspective

Jan Peleska
University of Bremen and Verified Systems International GmbH
peleska@uni-bremen.de

The 6th IFIP WG 1.8 Workshop on Trends in Concurrency Theory
September 9, 2017, Berlin, Germany

New Age Concurrency?

- Several observations lead us to the conviction that “time is right” to invest into changes of paradigm in the field of concurrency and its semantic foundations
  - Multi-core systems – the need for **weak memory models**
  - E-commerce – new notions of **distributed database consistency**
  - Cyber-physical systems (CPS) – **dynamic re-configuration, adaptive, emergent properties, collaborative, multi formalism development and V&V …**

Three Topics to Address

- **Multi-formalism support** for CPS modelling and verification
- **Dynamicity** – changing CPS configurations
- **Evolving behaviour** of CPS components

All this is presented from the perspective of **model-based testing**

Multi-formalism support for CPS modelling and verification

Problem Statement

- Different CPS components are developed and verified with **different formalisms**
- This produces **“local” verification results**, presented in different formalisms
- How can we assert the v**alidity of the required emergent properties** of the CPS?

Two Approaches

- Application of the
  - Theory of (Grothendiek) Institutions
  - Unifying Theory of Programming (UTP)

  to translate

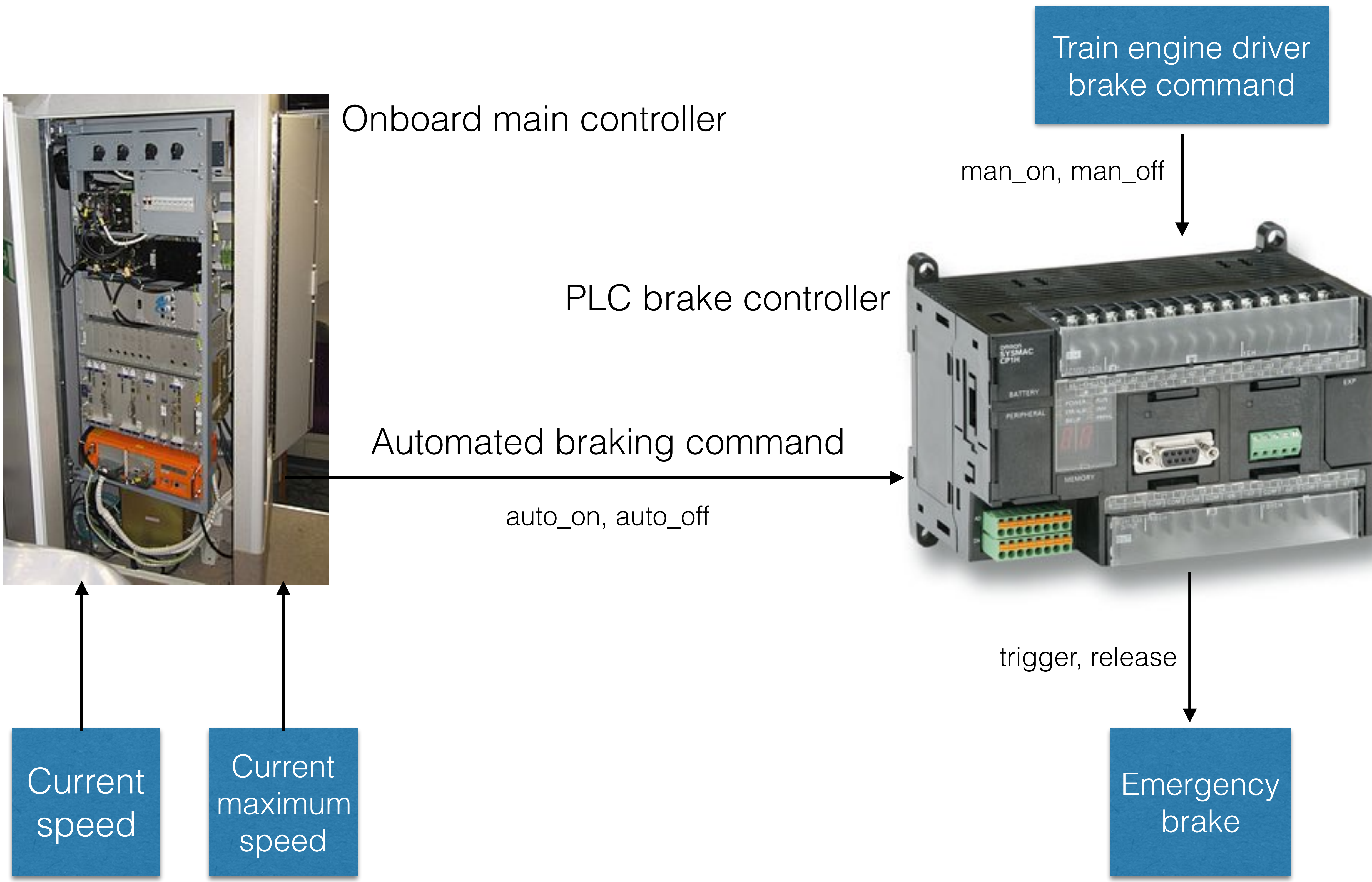
    - theories between different formalisms
    - verification obligations and test cases
    - verification results and test results

  between different formalisms

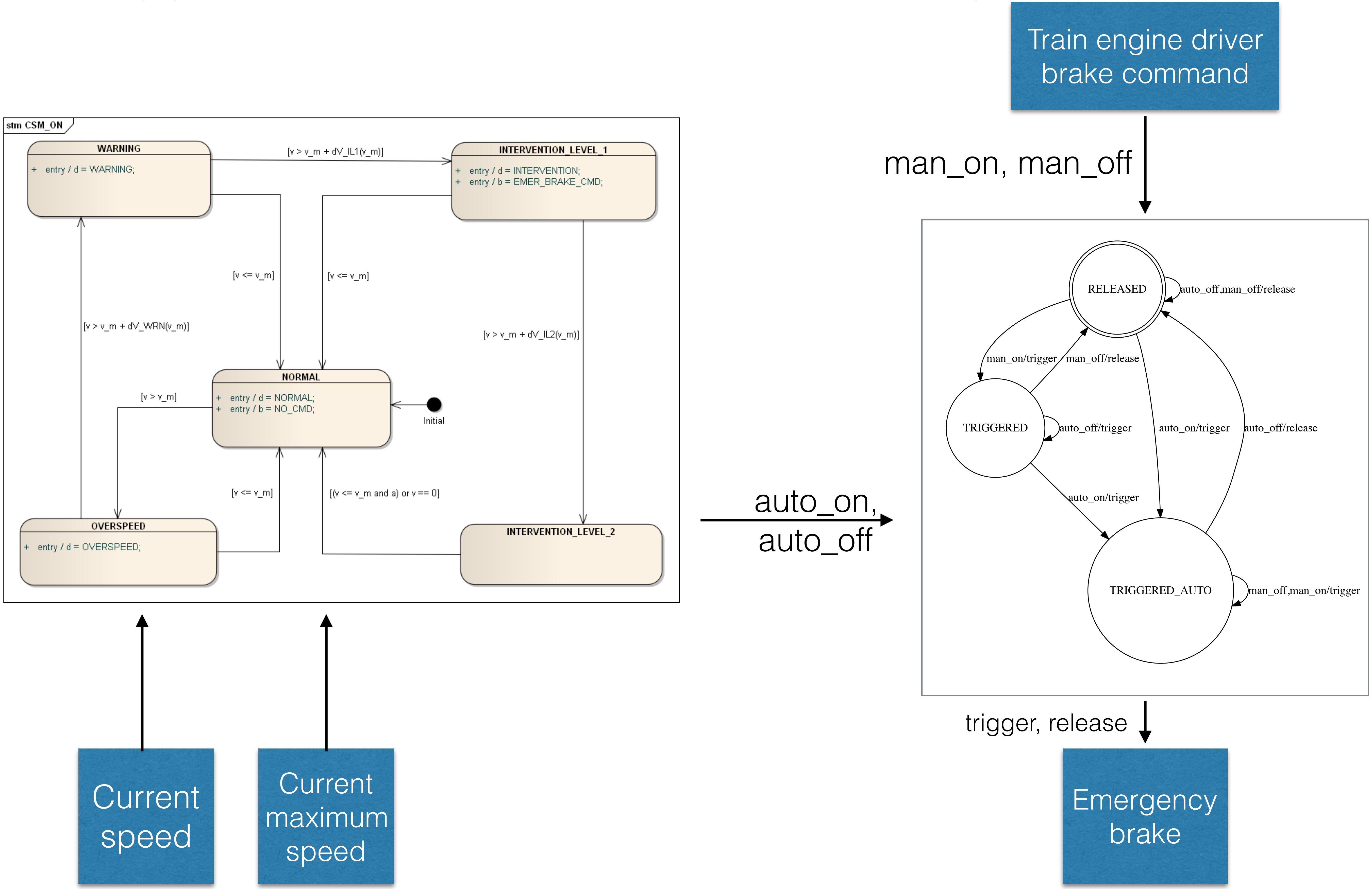
Application scenario

- CPS consists of several components
- Some components are modelled by finite state machines (FSMs)
- Other components are modelled by SysML state machines with Kripke structure semantics

Application scenario – train onboard speed control

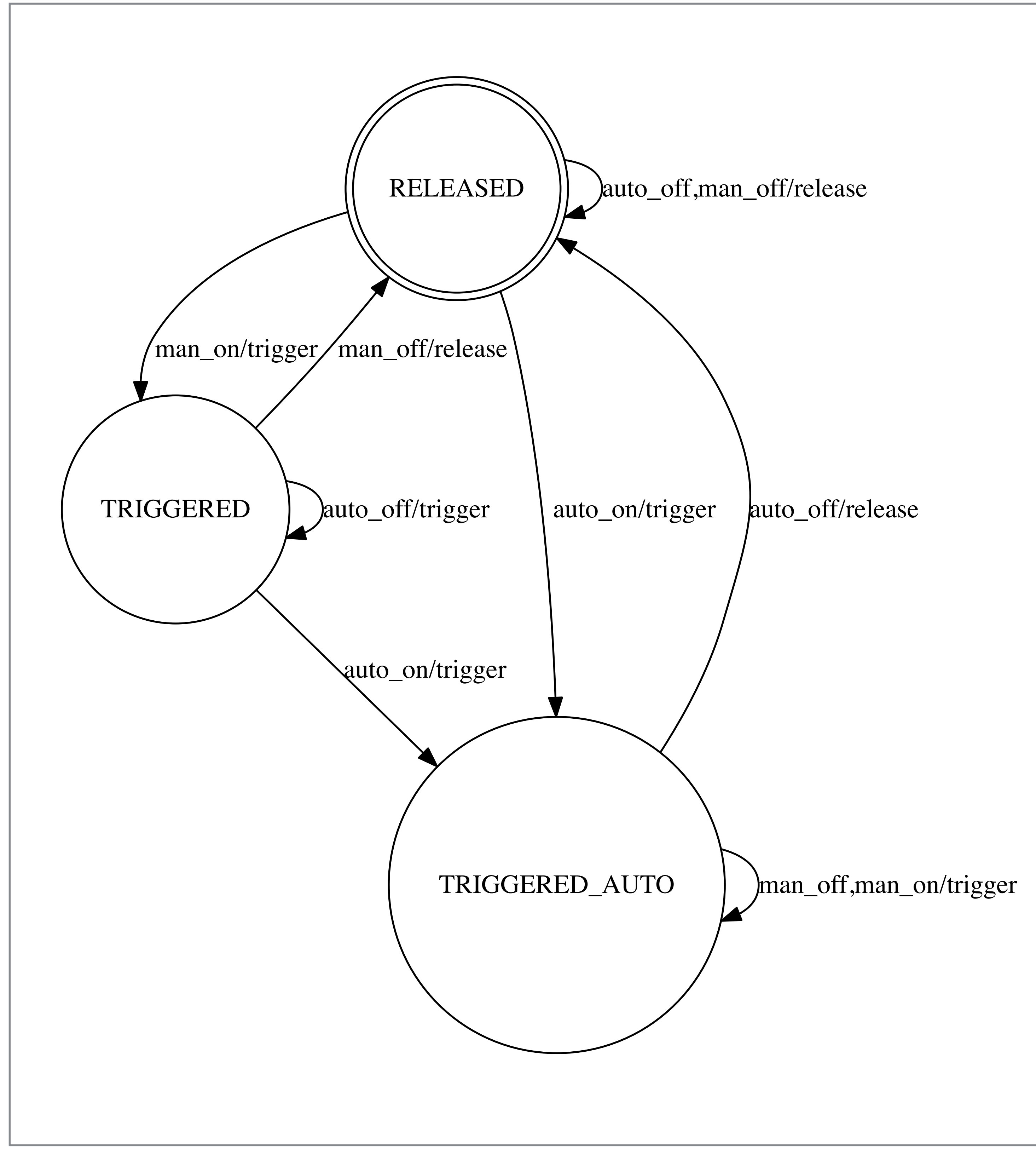


Application scenario – train onboard speed control



Brake controller

- Discrete inputs
- Discrete internal state
- Discrete outputs
- Complete testing strategies available



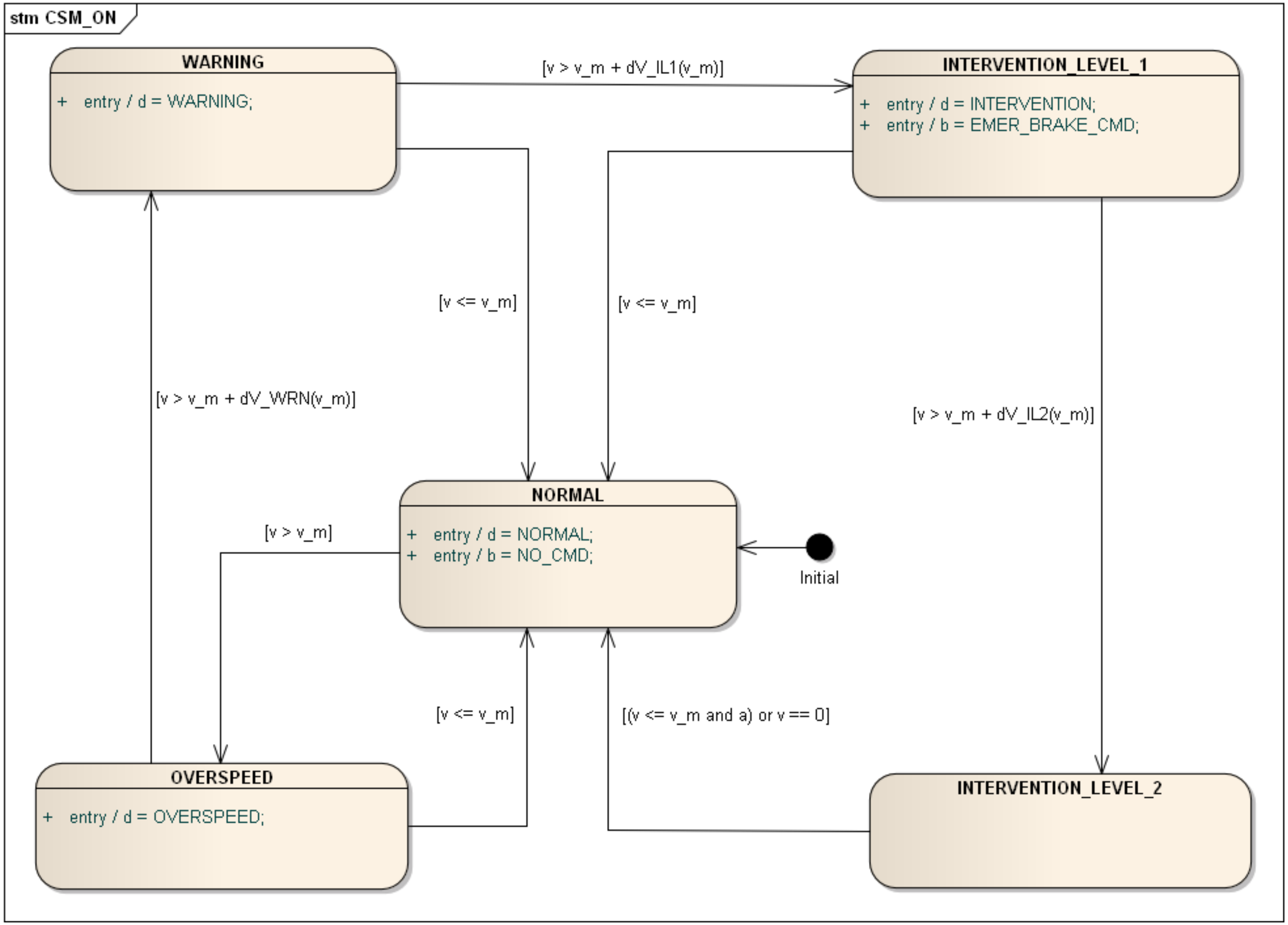
Onboard main controller

- Large input domains – speed
- Discrete internal state
- Discrete outputs

☆Apply input equivalence class testing

☆Can we also apply a complete strategy?

☆**TTT = Testing Theory Translation using institutions**



Verification of emergent properties

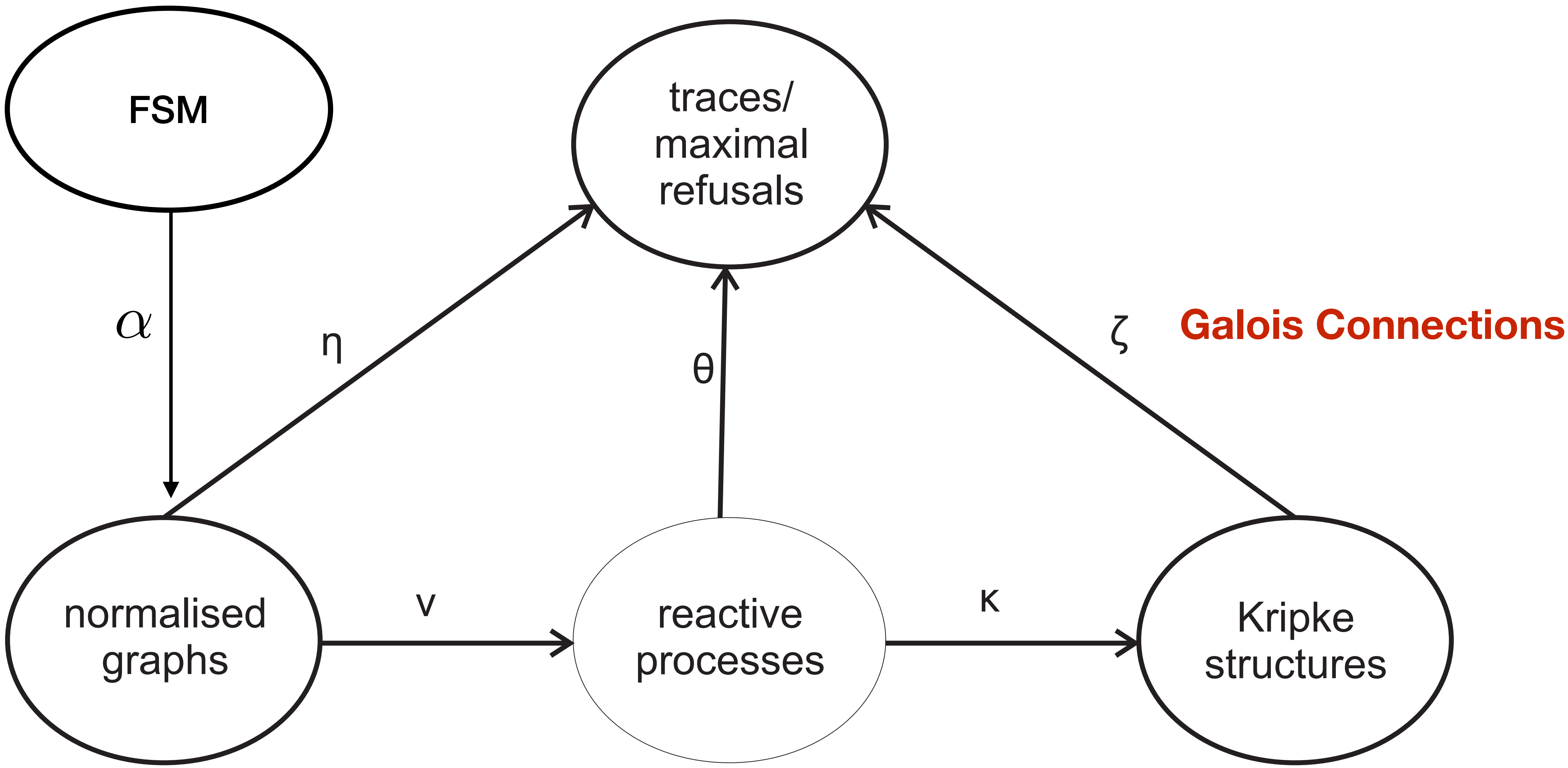
- **Application scenario**
    - Onboard controller has been verified and tested using SysML models with Kripke semantics
    - PLC has been verified and tested using FSM models
        - **Verification objective.** System satisfies emergent property

          EP. *„As long as the speed is above emergency threshold, the emergency brakes stay active and cannot be manually released“*
    - **Technical side condition.** EP shall be specified in CSP trace logic

Verification of emergent properties

- **Problems to be solved**
  - EP can only be specified by referring to properties of both the onboard main controller and the brake controller
  - Properties related to brake controller are specified by FSM I/O sequences **x/y** – e.g. via **intersection with testing automaton**
  - Properties related to Onboard speed controller are specified by, e.g. **LTL formulas with shared I/O variables** as free symbols
  - CSP trace logic formulas are specified over **traces of events and refusal sets**

Linking Theories by UTP



Ana Cavalcanti, Wen-ling Huang, Jan Peleska, **Jim Woodcock**:
CSP and Kripke Structures. ICTAC 2015: 505-523

Dynamicity – changing CPS configuration

Problem Statement

- CPS need cooperating components in **dynamically changing configurations**
- Each component needs to be prepared to
  - **accept/set up/destroy new communication links** from/to other components entering/leaving the configuration
  - enter/leave the configuration itself (**mobility**)

Major Contributions

- **pi-calculus** for dynamic creation of channels
- **Augmented CSP** allowing to simulate Pi-calculus with the means of a “conventional process algebra”
- **Bigraphs** for presenting both topography and communication structure

pi-Calculus and CSP

- Milner's pi-calculus

$$(\nu x)\big(\overline{x}\langle z\rangle.0 \mid x(y).\overline{y}\langle x\rangle.x(y).0\big) \mid z(v).\overline{v}\langle v\rangle.0$$

allows for **dynamic channel creation** and **communication of channel names**

- Roscoe showed that pi-calculus can be simulated by CSP augmented with **throw operator**

$$P \Theta_A Q$$

A.W.Roscoe:
CSP is Expressive Enough for pi.
C.B.Jones et. al. (eds.), Reflections on the work of C.A.R. Hoare, dog 10.1007/978-1-84882-912-1 16, Springer 2010

pi-Calculus and CSP

- Milner's pi-calculus

  $$(\nu x)(\overline{x}\langle z\rangle.0 \mid x$$

  allows for **dynami**
  communication o

As a consequence, "conventional" model checking (e.g. with FDR) can be used to verify mobile process systems

- Roscoe showed that pi-calculus can be simulated by CSP augmented with **throw operator**

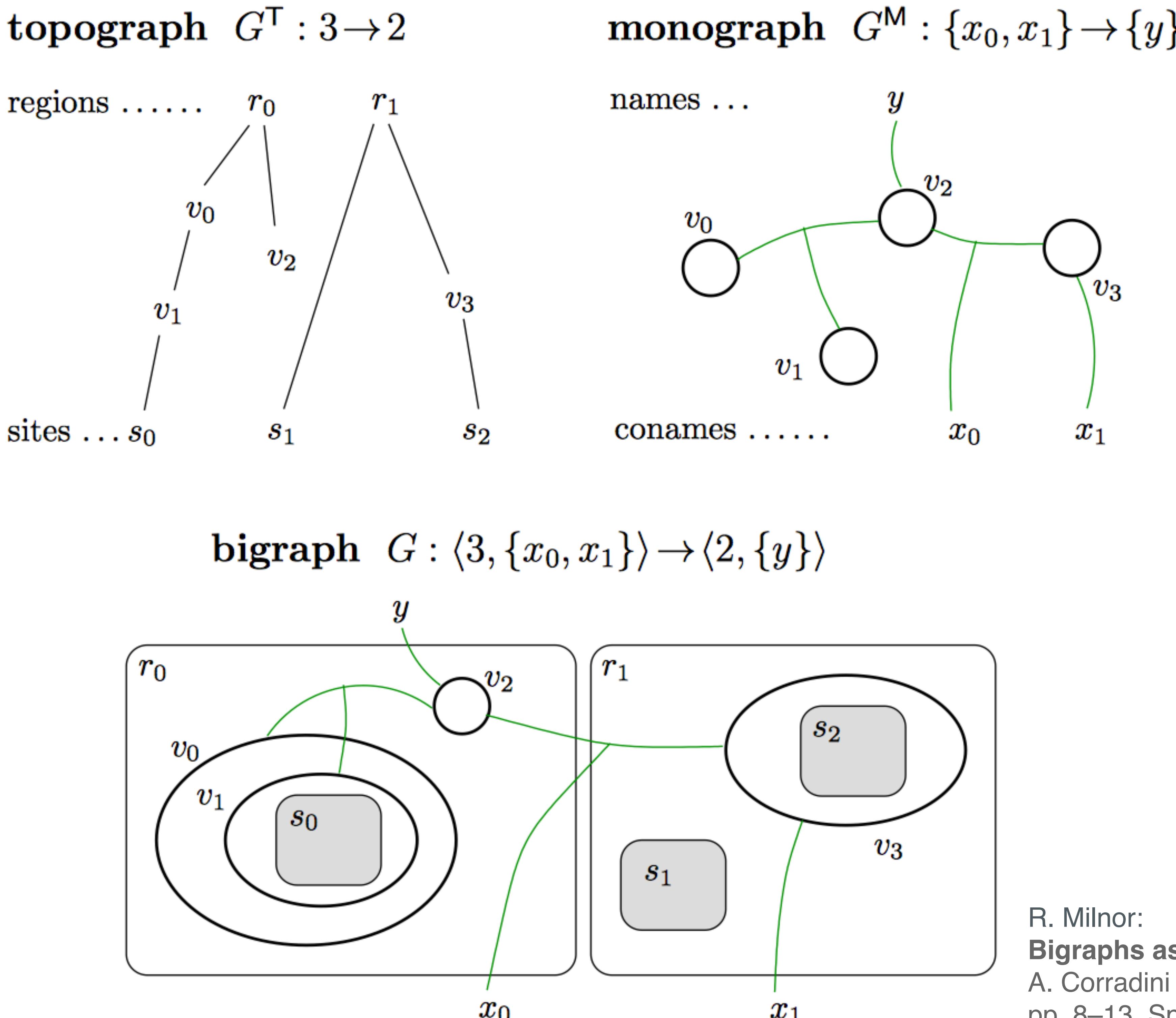
$$P \Theta_A Q$$

A.W.Roscoe:
CSP is Expressive Enough for pi.
C.B.Jones et. al. (eds.), Reflections on the work of C.A.R. Hoare, dog 10.1007/978-1-84882-912-1 16, Springer 2010

Bigraphs

- Bigraphs allow for representation of
  - **process topography**
  - **communication topology**
  - **dynamic changes** of the former

Bigraphs



R. Milnor:
Bigraphs as a Model for Mobile Interaction
A. Corradini et al. (Eds.): ICGT 2002, LNCS 2505, pp. 8–13, Springer 2002.

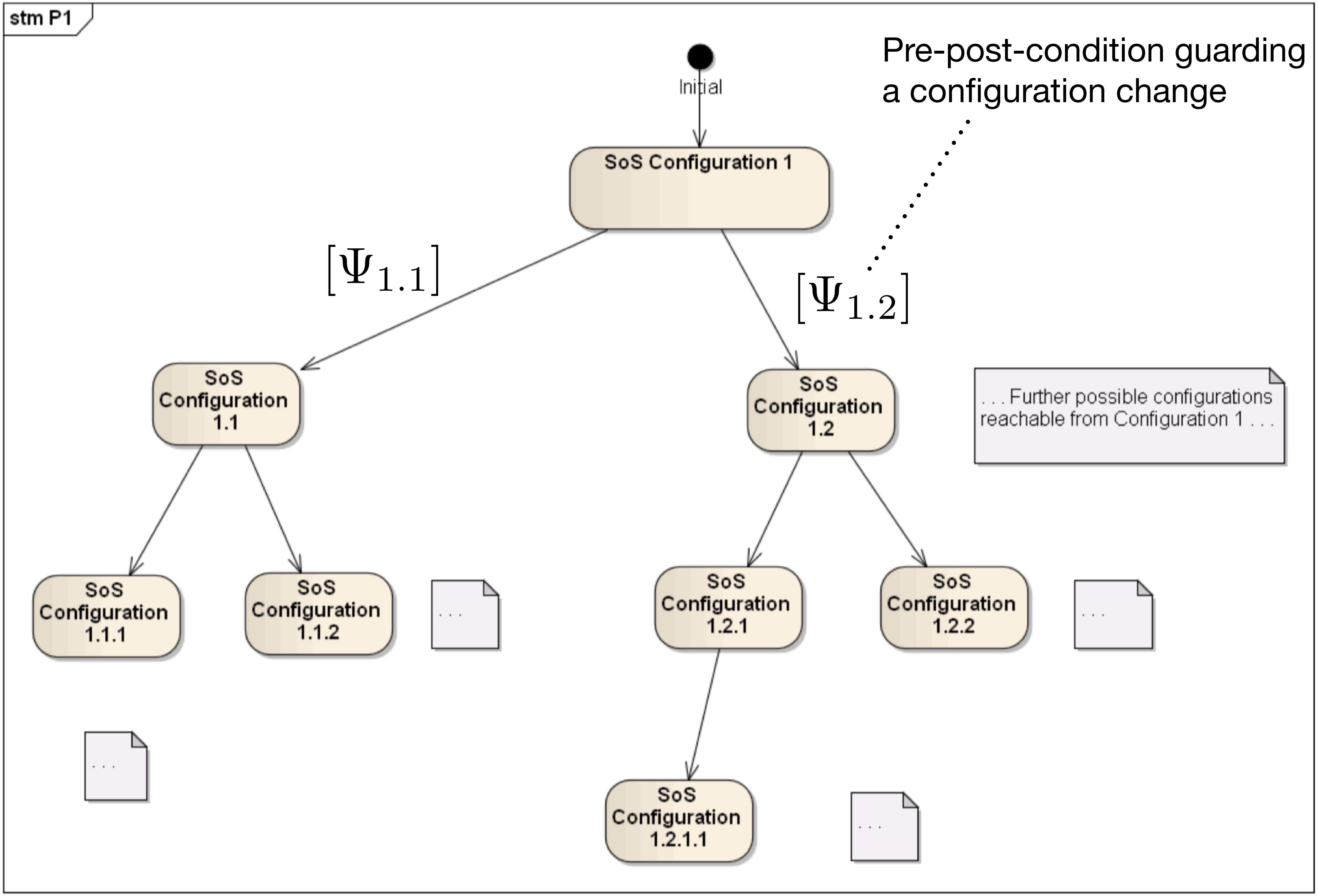
How to Test Dynamic CPS Configurations

- Some things are easier in testing than in general verification
  - Only **safety properties** matter
  - **Tests terminate** after finite amount of time
  - **Finite variability** of HW components implies that only a finite number of configurations can be covered during test execution

How to Test Dynamic CPS Configurations

- Model a **CPS configuration tree**
- Construct **equivalence classes** for configuration changes
- Elaborate **complete testing theory** guaranteeing full fault coverage with finitely many test cases, provided that
  - equivalence classes are adequate
  - CPS components do not have more state equivalence classes than assumed

CPS (or SoS) configuration tree



Evolving behaviour of CPS components

Problem Statement

- CPS components act according to the **rely-guarantee paradigm**

- The **assumptions** component C relies on **may be violated** after some time, due to

  - configuration changes

  - evolving behaviour of other components

- C needs to **adapt its behaviour** to the new environment conditions

What is to be Solved?

- **Detection**. Component needs to “understand” that its assumptions no longer hold
- **Change of belief**. Component needs to update its assumptions about the environment
- **Adaptation**. Component needs to “optimise” its behaviour w.r.t. the new assumptions

Detection

- For **regular safety properties**, the detection problem is completely solved
- Can be implemented efficiently for **hard real-time** applications

Recall. A safety property over atomic propositions AP is regular, if its **bad prefixes** in $(2^{AP})^*$ form a regular language.

Detection

- Therefore, the bad prefix set can be represented by **accepting states of an FSM**

- **One more problem to solve**. CPS component may not know the trace of system observations from the start, since it may join the configuration at a later state

    - Use a **homing algorithm** to determine the FSM state by a sequence of observations

- The detection problem is a **passive testing problem**

Detection – an Example

- Suppose, component C relies on the environment to fulfil safety condition

$$\begin{aligned}\Phi \quad \equiv \quad & s_0 \wedge \mathbf{G}\Big((s_0 \wedge \mathbf{X}(s_1 \wedge a)) \vee \\ & (s_1 \wedge \mathbf{X}((s_0 \wedge b) \vee (s_2 \wedge a))) \vee \\ & (s_2 \wedge \mathbf{X}(s_1 \wedge b))\Big)\end{aligned}$$

with internal state variable s_j and assumption that a or b must occur in every step

Example trace. $a.b.a.a.b.a.b.b\ldots$

Detection

b follows a – at most two more a’s than b’s

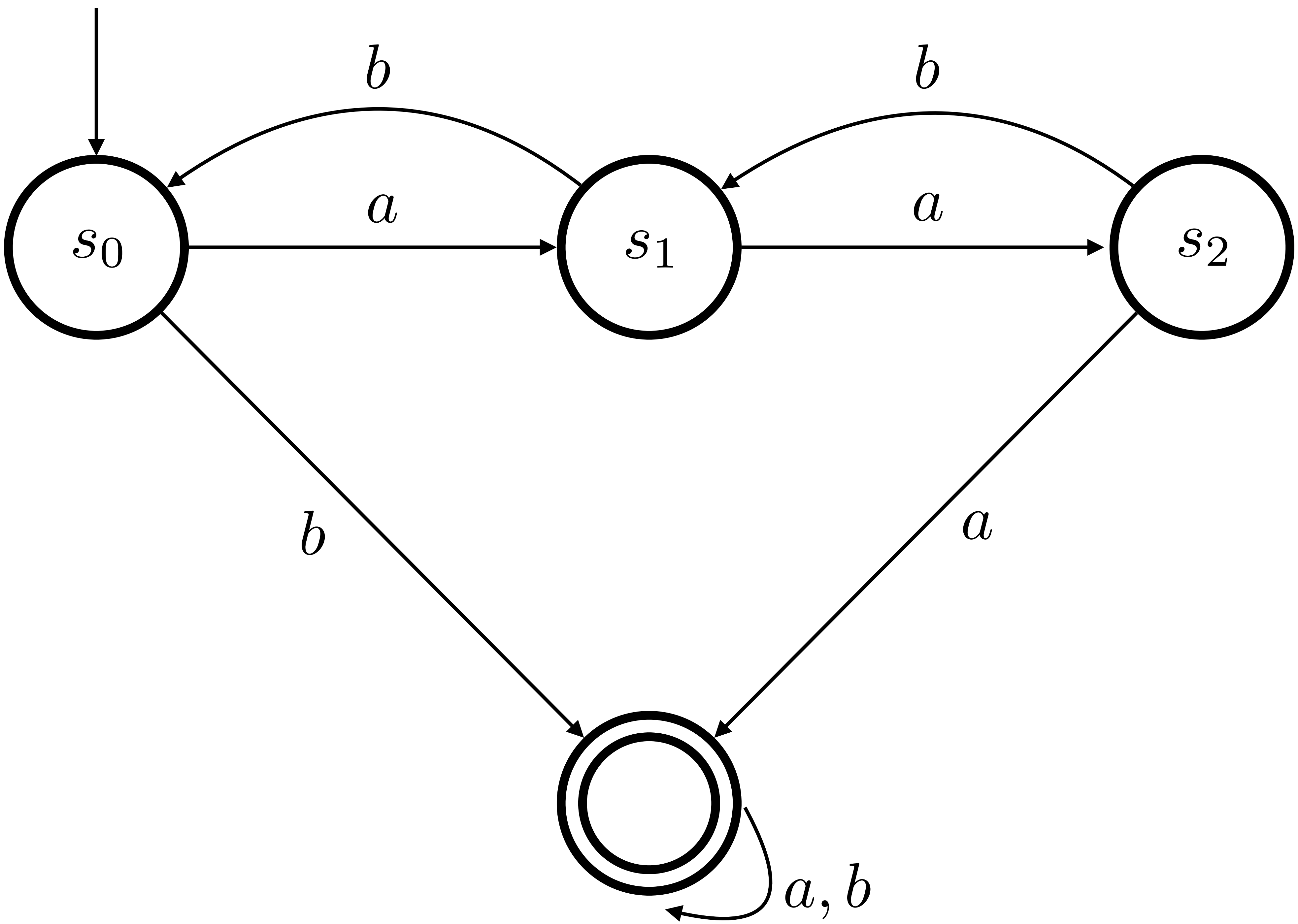
- Suppose, component C
  safety condition

$$\Phi \equiv s_0 \wedge \mathbf{G}\Big((s_0 \wedge \mathbf{X}(s_1 \wedge a)) \vee (s_1 \wedge \mathbf{X}((s_0 \wedge b) \vee (s_2 \wedge a))) \vee (s_2 \wedge \mathbf{X}(s_1 \wedge b))\Big)$$

with internal state variable s_j and assumption that a or b must occur in every step

Example trace. $a.b.a.a.b.a.b.b\ldots$

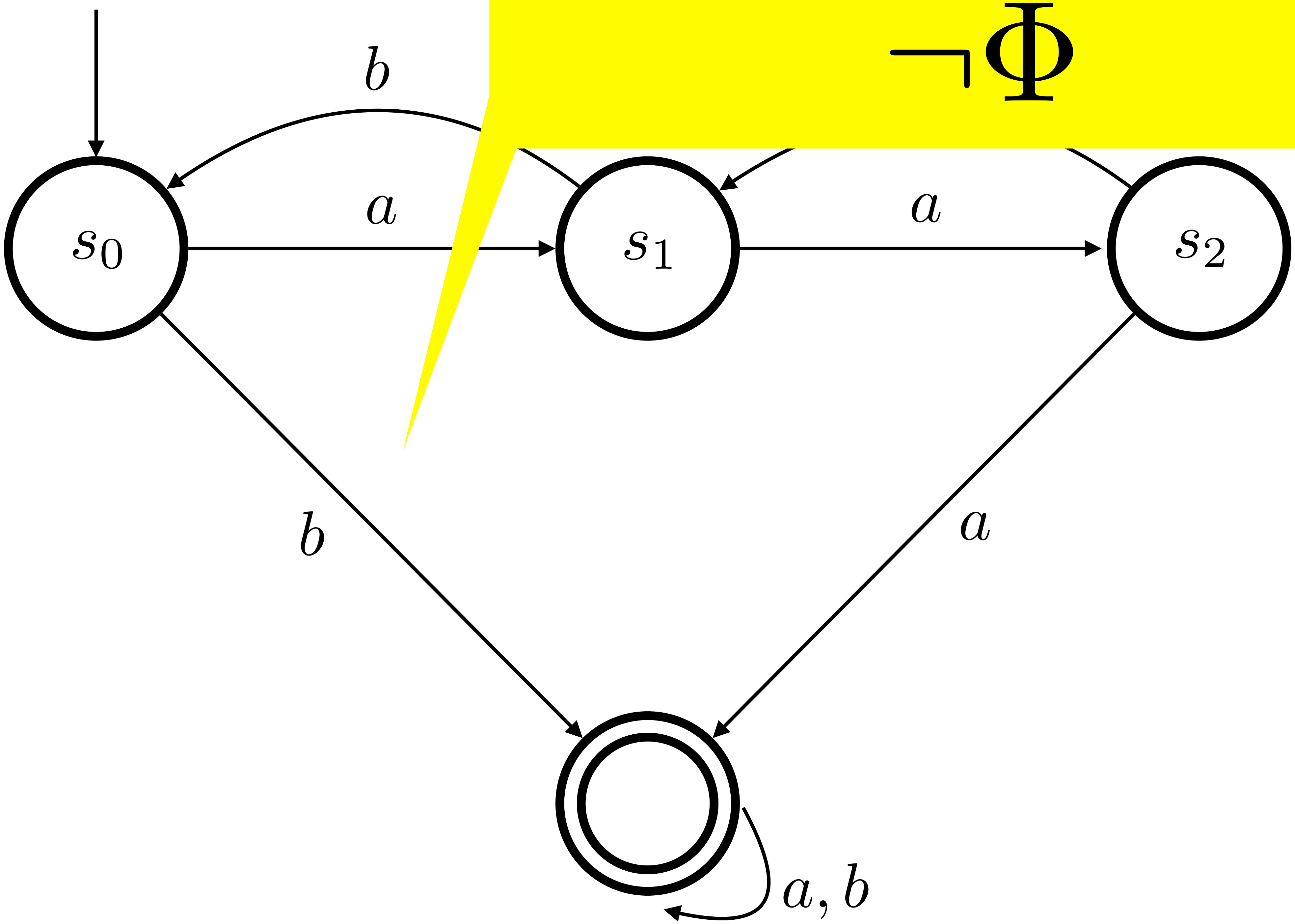
Example (continued). FSM modelling bad prefixes of the safety condition



Example (continued). condition

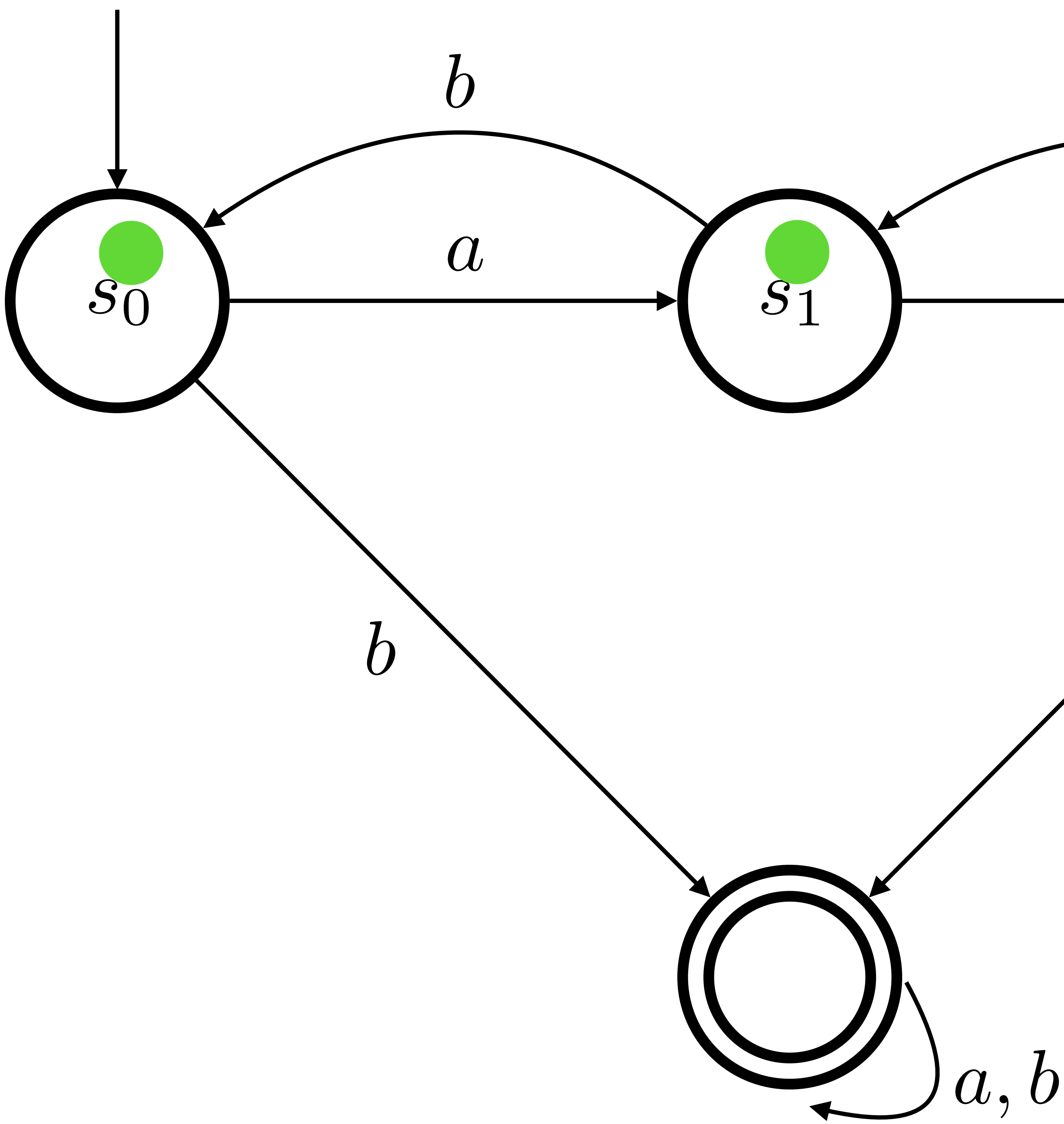
Can be generated, for example with ltl2ba from

$$\neg\Phi$$



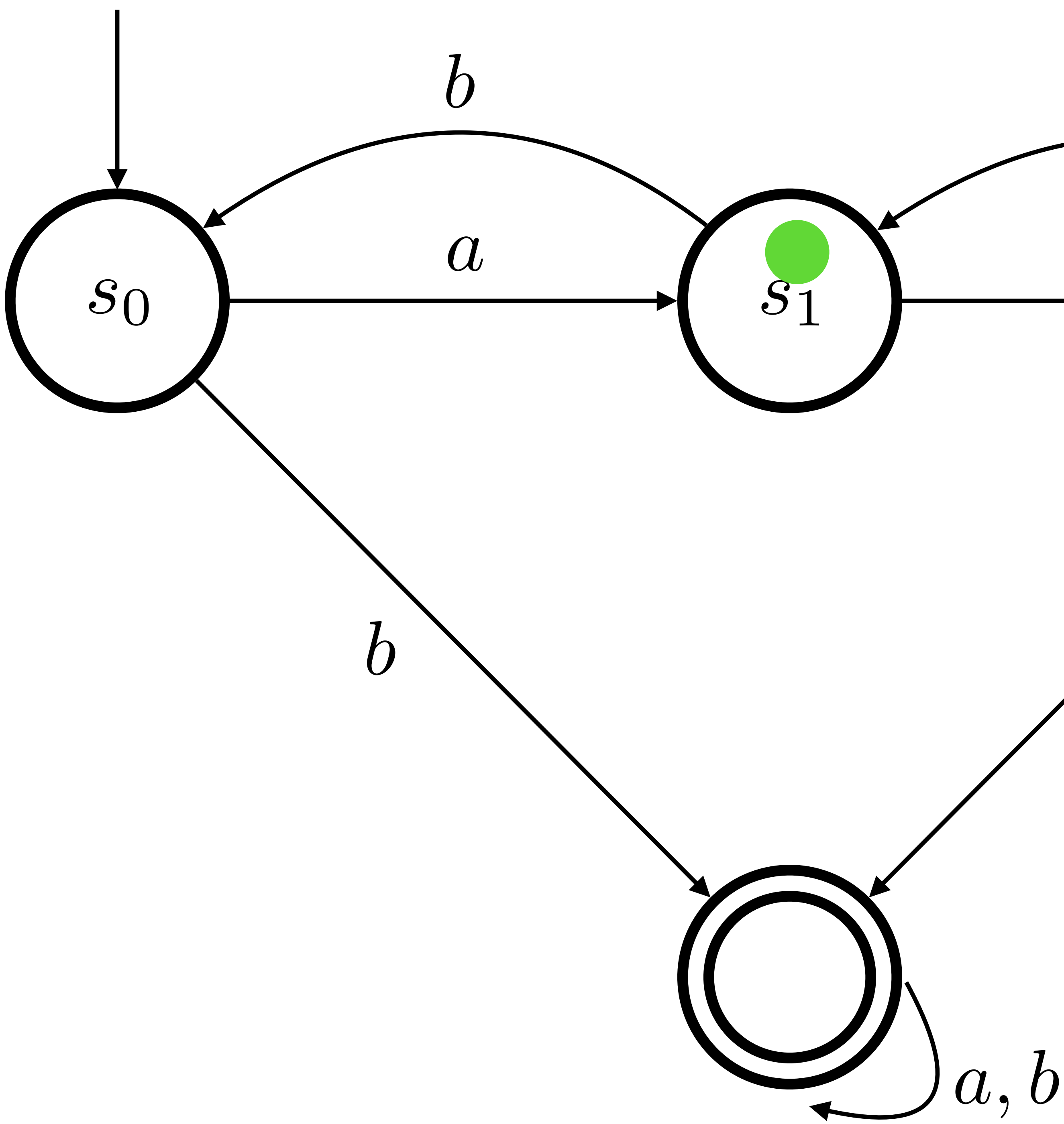
Example (continued). Application of the homing algorithm

Initial checking state



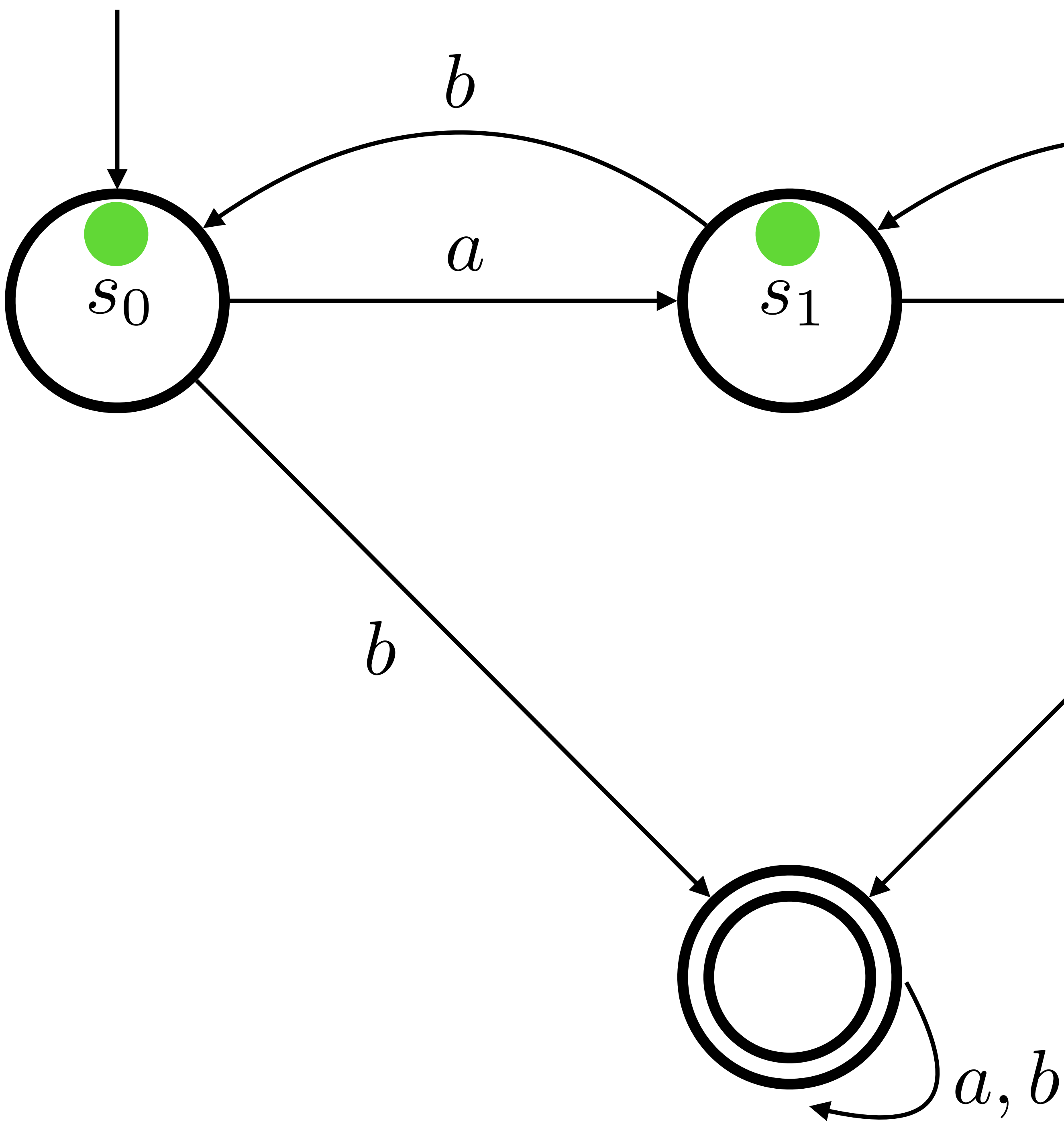
Example (continued). Application of the homing algorithm

Observation b



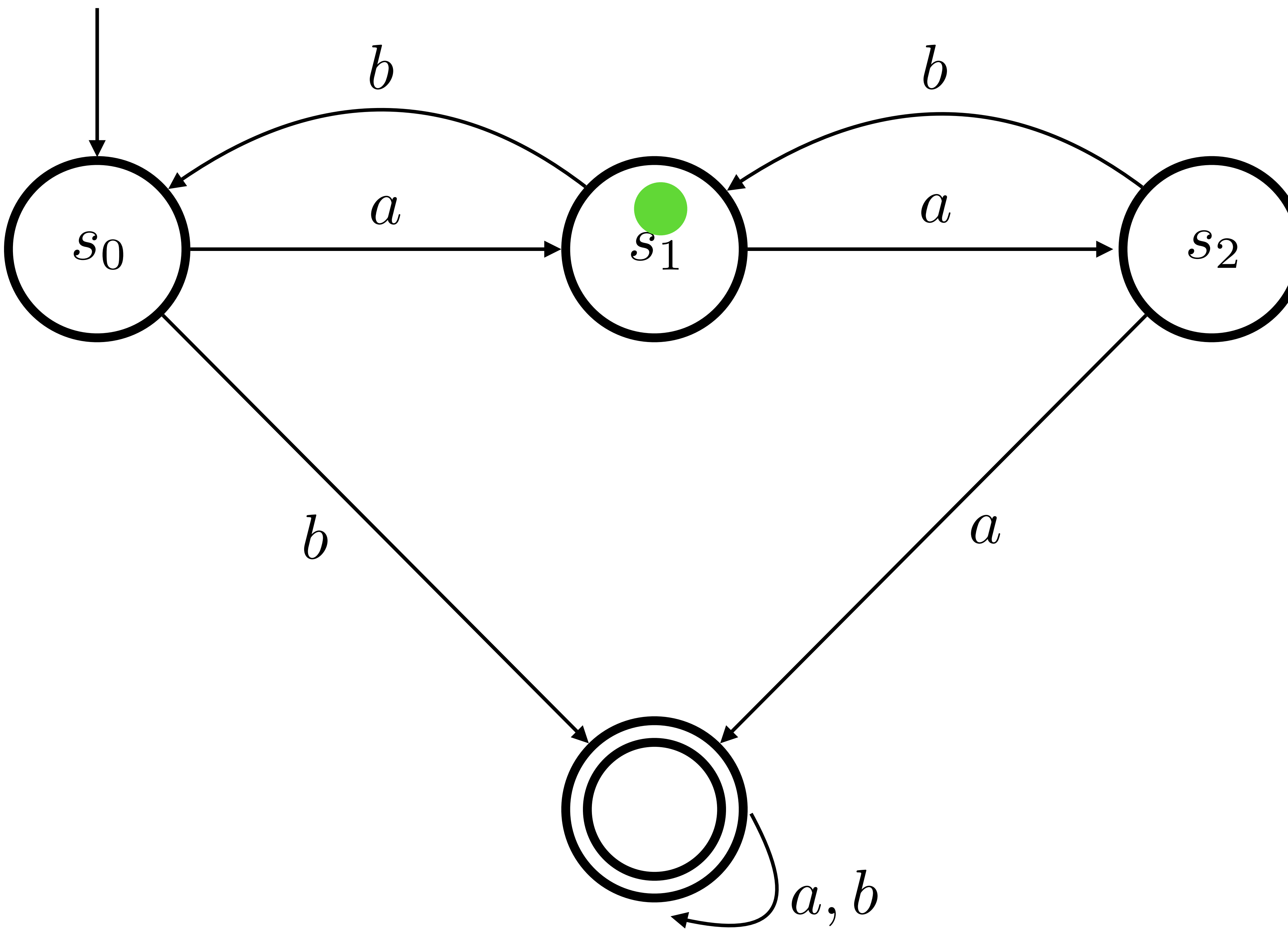
Example (continued). Application of the homing algorithm

Observation b – post-state



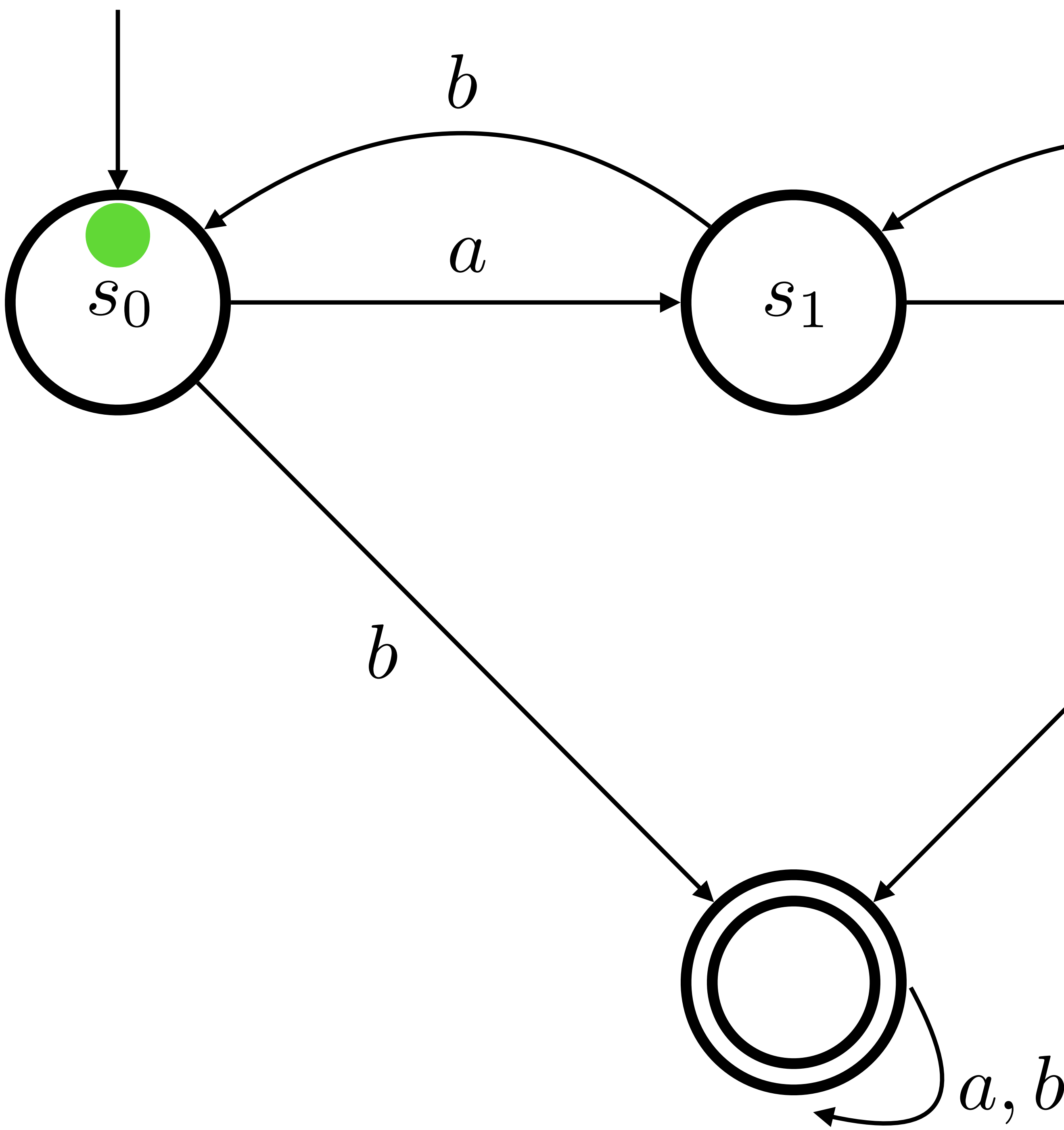
Example (continued). Application of the homing algorithm

Observation b.b



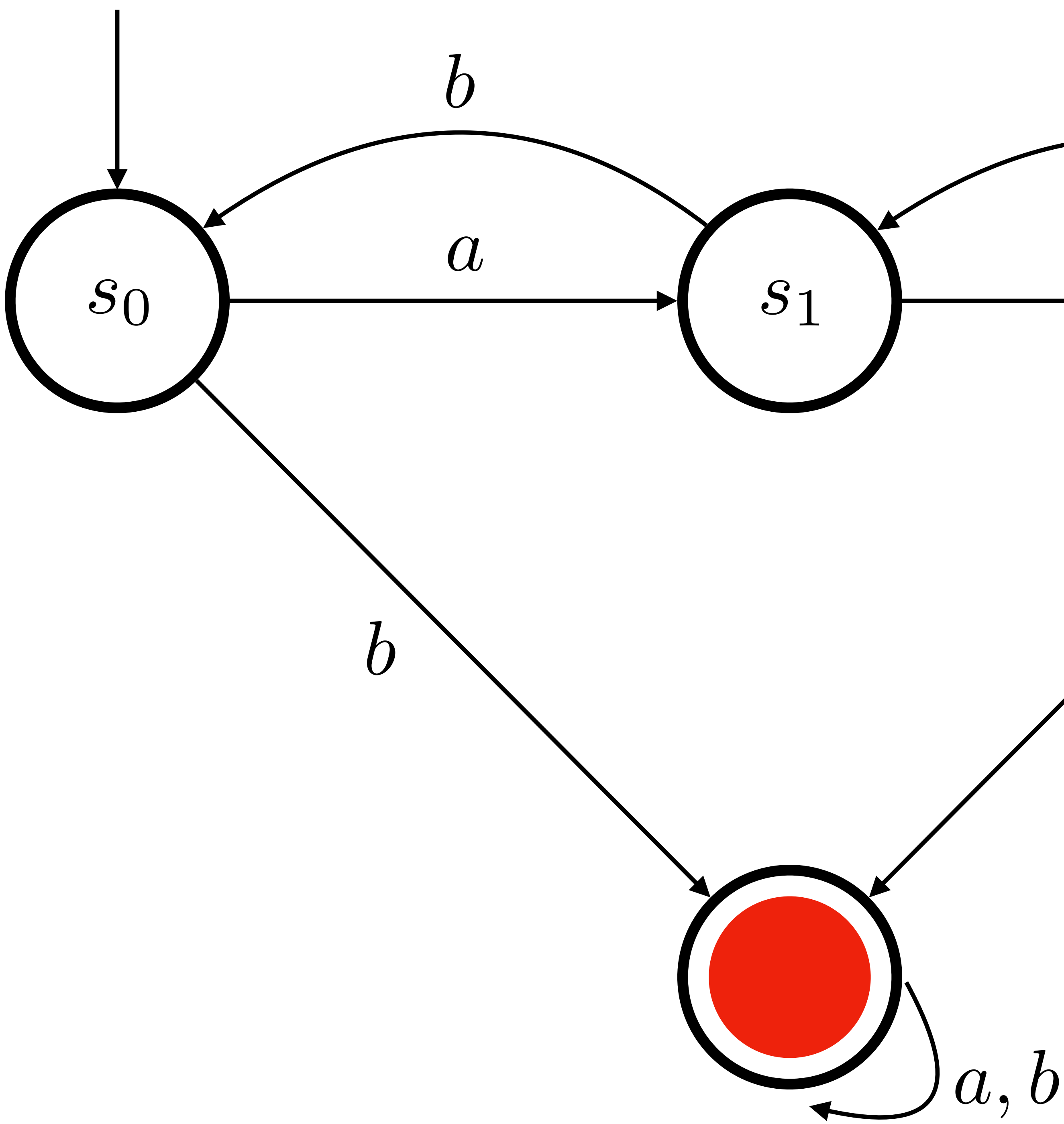
Example (continued). Application of the homing algorithm

Observation b.b – post-state



Example (continued). Application of the homing algorithm

Observation b.b.b – safety-violation



Change of Belief

- Different options with different complexity
  - **Assumptions are just “true”** – components can adapt to any environment behaviour (examples from control theory)
  - **Expected violations** of assumptions – **fault-tolerant adaptation** of behaviour under new, pre-defined assumptions
  - **Unexpected violations** of assumptions – new valid assumptions need to be extracted from observations (apply **machine learning**, construct temporal properties reflecting environment behaviour)

Change of Belief

- Different options with different complexity
  - **Assumptions are ju** any environment bel
  - **Expected violations** **adaptation** of behav assumptions
  - **Unexpected violations** of assumptions – new valid assumptions need to be extracted from observations (apply **machine learning**, construct temporal properties reflecting environment behaviour)

Can this be achieved in hard real-time?

Adaptation

- Solved, as far as
  - basic laws of **control theory** can be applied
  - optimal behaviour can be specified as mathematical boundary value problem or general optimisation problem
- Can be modelled by **Hybrid Automata**, if discrete changes between different control laws/optimisation methods are required

Adaptation

- Open questions
  - **Q1.** After change of belief system consisting of (temporal) logic formulas: how can we defined the **optimal behaviour** w.r.t. goals and belief system ?
  - **Q2.** If such a temporal logic formula for optimal behaviour could be found, could it become possible to **synthesise** the new component behaviour on the fly in hard real-time?

A Tentative Solution for Q1

- Specialised problem statement
  - If, due to changes in the environment behaviour, a CPS component can no longer fulfil its original guarantees, is there a possibility to specify a graceful degradation of behaviour in a well-founded way?
- Suggestion from testing theory
  - Classify component outputs according to criticality
  - Identify outputs of “negligible” criticality
  - Realise behaviour that is equivalent to the original specification, with all outputs of negligible criticality identified

Conclusion

- We discussed 3 topics of new-age concurrency
  - Multi-formalism support for CPS development and verification
  - Modelling and testing of dynamically changing CPS configurations
  - Modelling and testing evolving behaviour of CPS
- We have seen that many “mechanisms” and approaches already exist to tackle these challenges
- Do we need more — a comprehensive new theory & formalism, instead of a “bag of special solutions” ?

Further Reading

- About cyber-physical systems and mobile and channels

A.W. Roscoe. CSP is Expressive Enough for π. In C.B. Jones et al. (eds.), Reflections on the Work of C.A.R. Hoare, DOI 10.1007/978-1-84882-912-1 16, Springer, 2010.

Jim Woodcock, Andy Wellings, and Ana Cavalcanti. Mobile CSP. In M. Cornelio and B. Roscoe (Eds.): SBMF 2015, LNCS 9526, pp. 39–55, 2016. DOI: 10.1007/978-3-319-29473-5 3, Springer, 2016.

- About testing and equivalence classes

Wen-ling Huang and Jan Peleska. Complete model-based equivalence class testing for nondeterministic systems. DOI 10.1007/s00165-016-0402-2 BCS © 2016 Formal Aspects of Computing (2017) 29: 335–364

Acknowledgements

I would like to thank Mohammad Reza Mousavi and all organisers of the IFIP 1.8 workshop for the invitation to present this keynote.

Special thanks go to Ana Cavalcanti and Jim Woodcock for pointing out to me some crucial facts about mobile system and their semantics.

The material about testing presented here is based on joint work with Wen-ling Huang.