Semantic Families for Cyber-physical Systems

Jan Peleska
University of Bremen
Verified Systems International GmbH
jp@cs.uni-bremen.de
2015-12-07

BCS FACS - Annual Peter Landin Semantics Seminar 2015

Overview

- Semantics for CPS – time for a change of paradigm?
- Multiple formalisms in CPS modelling
  - Example 1. Testing theories and collaborative tool environments
  - Example 2. Verification of emergent properties
- Conclusions and future work

- **Semantics for CPS – time for a change of paradigm?**
- Multiple formalisms in CPS modelling
    - Example 1. Testing theories and collaborative tool environments
    - Example 2. Verification of emergent properties
- Conclusions and future work

Semantics for CPS – time for a change of paradigm ?

Recall

- The investigation of concurrent systems semantics started somewhere in the seventies of the last century

C. A. R. Hoare:
Communicating Sequential Processes. Commun. ACM 21(8): 666-677 (1978)

Recall

- Since then, a multitude of formalisms has been developed and successfully applied to

- **Development**
  - modelling
  - code generation

- **Verification & Validation**
  - theorem proving
  - model checking
  - simulation
  - testing

Cyber-physical systems

- Systems of collaborating computational elements controlling physical entities

https://en.wikipedia.org/wiki/Cyber-physical_system

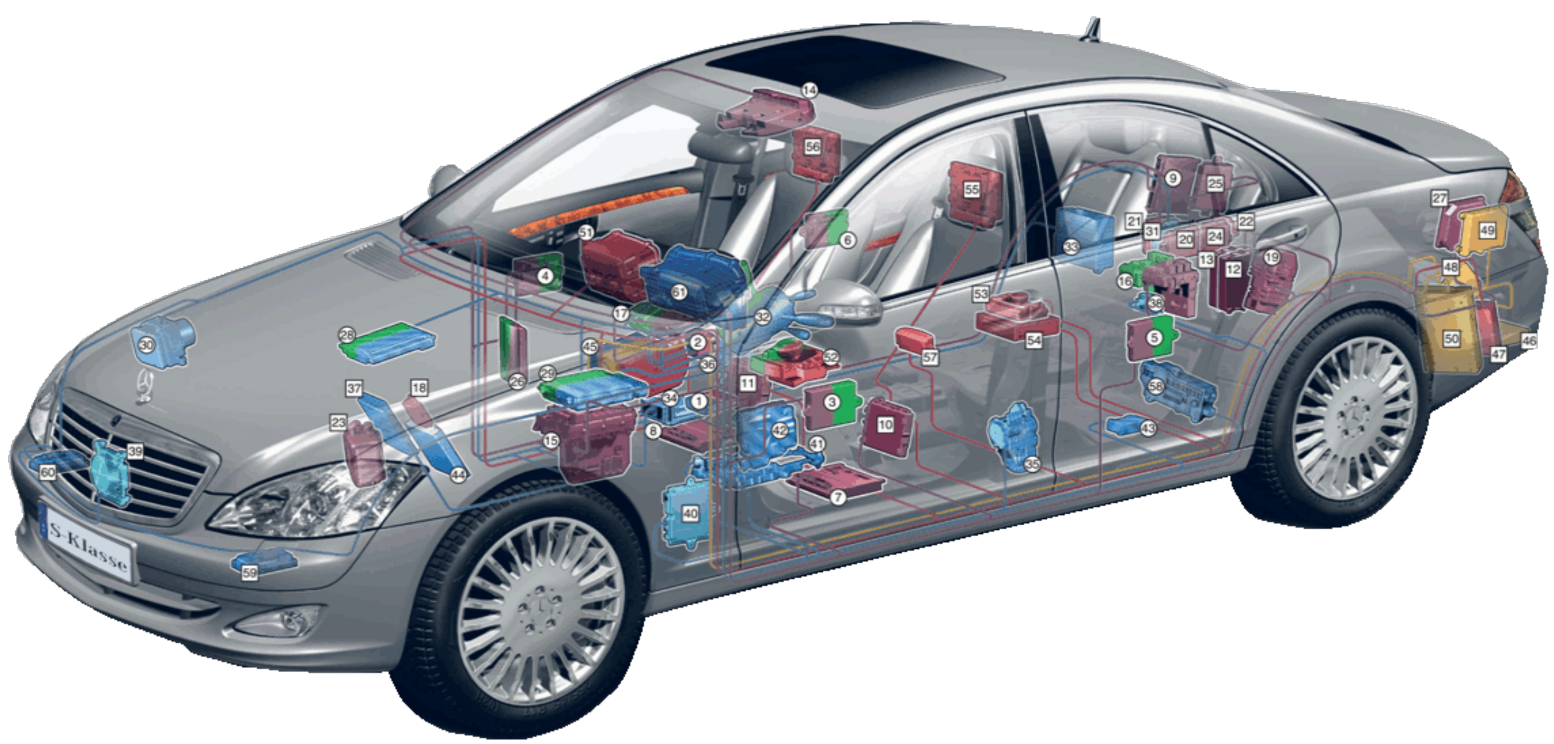


Image courtesy of Daimler AG

Some CPS-characteristics affecting semantic modelling

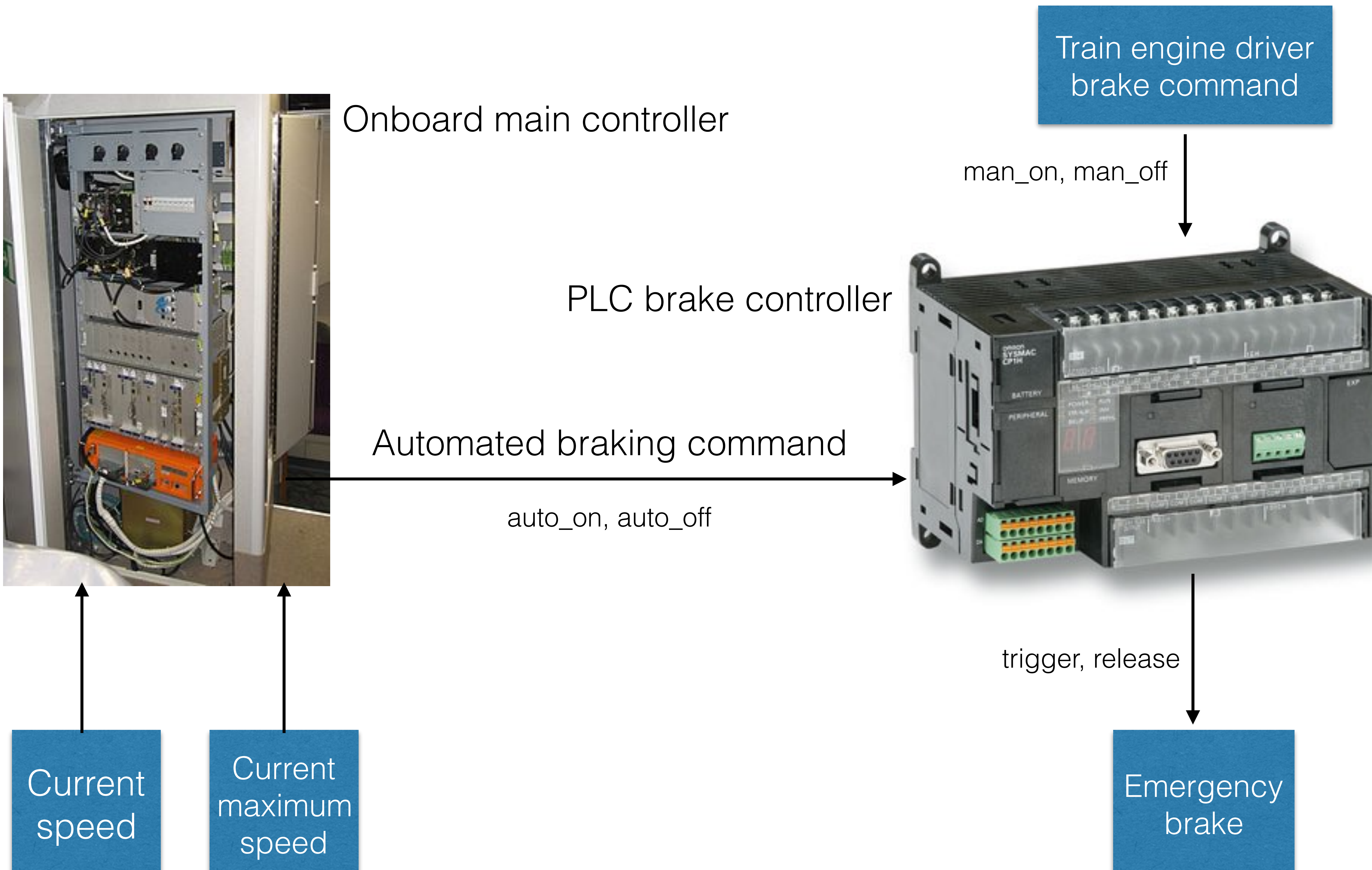
Characteristic	Semantics
Distribution, time-discrete and time-continuous control	☑ Hybrid systems semantics
Modeling using multiple formalisms	**Model, sentence, and theory translation**
Emergent properties	Temporal logic, trace logic – **how to verify in presence of multiple formalisms?**
Dynamic re-configuration	Semantics for object-oriented systems – or can we find something simpler?
Evolution of asserted component behaviours	New paradigms for behavioural assertions?
Large numbers of replicated components	Can the knowledge about replication lead to optimised V&V methods?

Multiple formalisms in CPS modelling – Example 1. Testing theories and collaborative tool environments

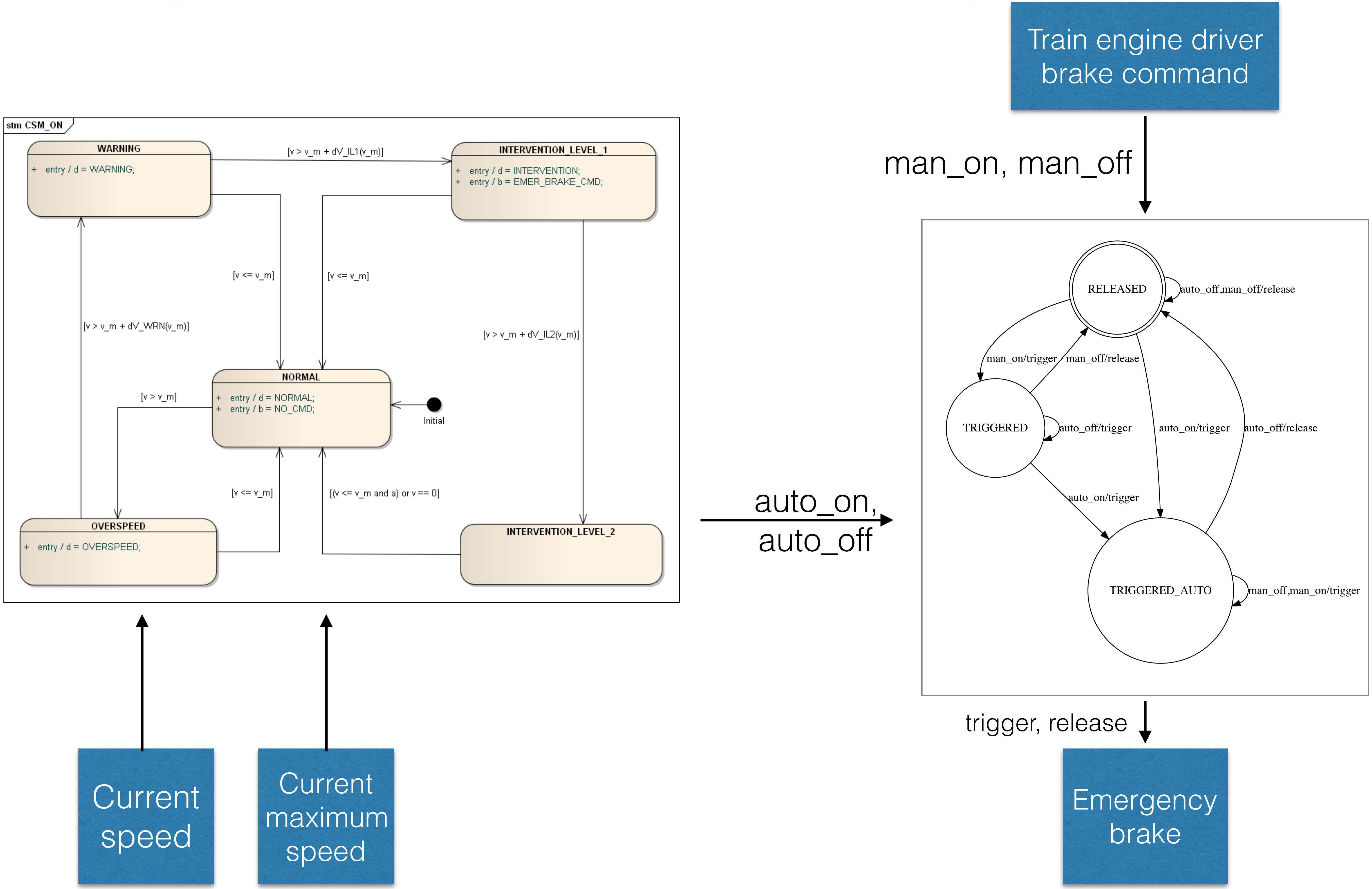
Application scenario

- 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



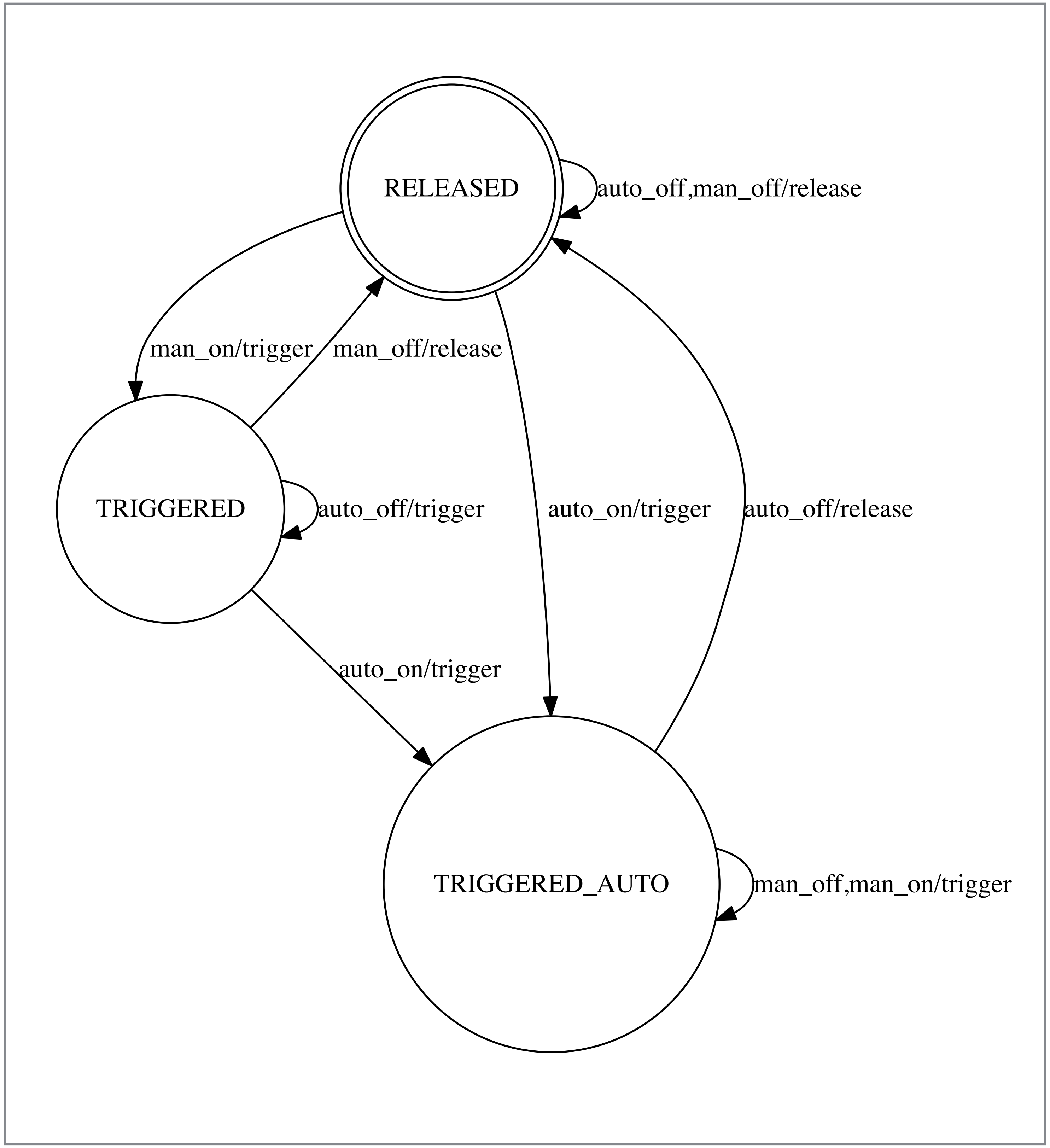
Application scenario – train onboard speed control



- Discrete inputs
- Discrete internal state
- Discrete outputs

⭐Apply **complete FSM testing strategy**

Brake controller



Complete test suites

- Defined with respect to **fault model** (*M*, ≤, *Dom*), that is,
  - a reference model *M*
  - a conformance relation ≤
  - a fault domain *Dom*
- **Complete** = sound + exhaustive
- **Sound** = every *M'* in *Dom* <u>satisfying</u> *M'* ≤ *M* <u>passes</u>
- **Exhaustive** = every *M'* in *Dom* <u>violating</u> *M'* ≤ *M* <u>fails</u>

Complete FSM test suites

- For FSMs, many complete testing strategies exist
  - for deterministic or nondeterministic FSMs
  - for completely defined or incomplete FSMs

Alexandre Petrenko, Nina Yevtushenko:
Adaptive Testing of Nondeterministic Systems with FSM. HASE 2014: 224-228

Robert M. Hierons:
Testing from a Nondeterministic Finite State Machine Using Adaptive State Counting.
IEEE Trans. Computers 53(10): 1330-1342 (2004)

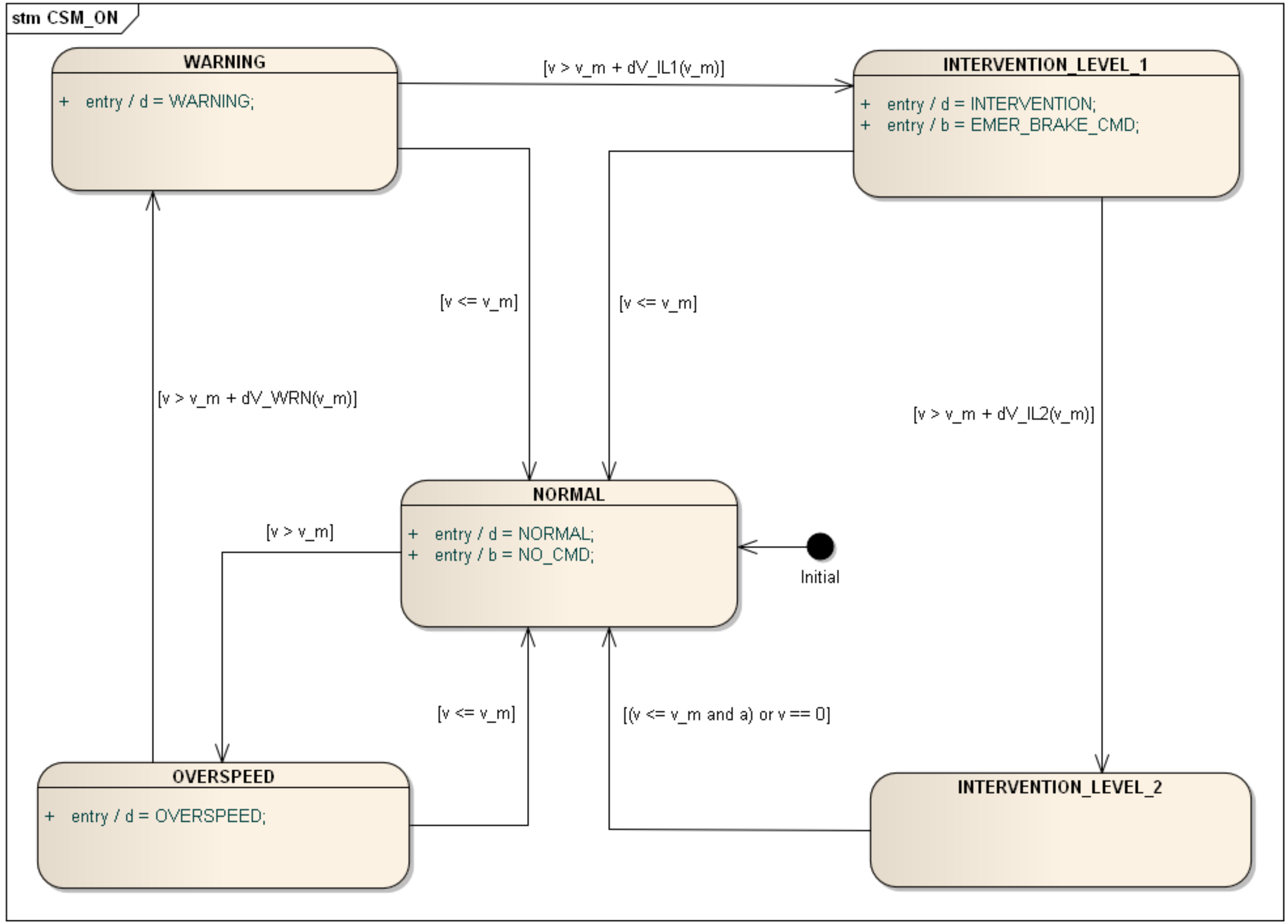
Onboard main controller

- 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**



TTT

- Consider different **semantic domains** with their conformance relations
  - Finite state machines – language equivalence, language containment
  - Kripke structures – I/O-equivalence, I/O-refinement
- Fix a **signature** in each domain
  - Sig_1 – Kripke structures over fixed I/O variables
  - Sig_2 – FSMs over fixed I/O-alphabet

TTT

- Create a **model map** *T* from sub-domain of *Sig*$_1$ to *Sig*$_2$

$$T : Dom_1 \to Sig_2;$$
$$Dom_1 \subseteq Sig_1$$

- Create a **test case map** T* from test cases of *Sig*$_2$ to test cases of *Sig*$_1$

$$T^* : \mathrm{TC}(Sig_2) \to \mathrm{TC}(Sig_1)$$

- Prove the **satisfaction condition**

Satisfaction condition

Condition 1. The model map is compatible with the conformance relations

$$\forall \mathcal{S}, \mathcal{S}' \in Dom_1 : \mathcal{S}' \leq_1 \mathcal{S} \Leftrightarrow T(\mathcal{S}') \leq_2 T(\mathcal{S})$$

Condition 2. Model map and test case map preserve the *pass* relationship

$$\forall \mathcal{S} \in Dom_1, U \in \mathrm{TC}(Sig_2) : T(\mathcal{S}) \ \underline{\mathrm{pass}_2}\ U \Leftrightarrow \mathcal{S}\ \underline{\mathrm{pass}_1}\ T^*(U)$$

Satisfaction condition,

reflected by commuting diagrams and relational composition

Condition 1

$$
\begin{tikzcd}
T(Dom_1) \arrow[r, "\leq_2"] & T(Dom_1) \\
Dom_1 \arrow[u, "T"] \arrow[r, "\leq_1"] & Dom_1 \arrow[u, "T"']
\end{tikzcd}
$$

$$T; \leq_2 \; = \; \leq_1; T$$

Condition 2

$$
\begin{tikzcd}
T(Dom_1) \arrow[r, "\underline{\text{pass}}_2"] & \mathrm{TC}(Sig_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"] \arrow[r, "\underline{\text{pass}}_1"] & T^*(\mathrm{TC}(Sig_2))
\end{tikzcd}
$$

$$\underline{\text{pass}}_1 \;=\; T; \underline{\text{pass}}_2; T^*$$

Recall. Relational composition

$$f \subseteq X \times Y, \quad g \subseteq Y \times Z$$

$$f ; g = g \circ f = \{(x, z) \mid \exists y : (x, y) \in f \wedge (y, z) \in g\}$$

Condition 1

$$\begin{tikzcd}
T(Dom_1) \arrow[r, "\leq_2"] & T(Dom_1) \\
Dom_1 \arrow[u, "T"] \arrow[r, "\leq_1"] & Dom_1 \arrow[u, "T"']
\end{tikzcd}$$

$$T ; \leq_2 \; = \; \leq_1 ; T$$

Condition 2

$$\begin{tikzcd}
T(Dom_1) \arrow[r, "\underline{\mathrm{pass}}_2"] & \mathrm{TC}(Sig_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"] \arrow[r, "\underline{\mathrm{pass}}_1"] & T^*(\mathrm{TC}(Sig_2))
\end{tikzcd}$$

$$\underline{\mathrm{pass}}_1 \;\; = \;\; T ; \underline{\mathrm{pass}}_2 ; T^*$$

General theorem for translation of testing theories

Theorem 1. Suppose *(T,T*)* exist and fulfil the satisfaction condition. Then every complete (sound, exhaustive) testing theory established in Sig_2 induces a likewise complete (sound, exhaustive) testing theory on Sig_1

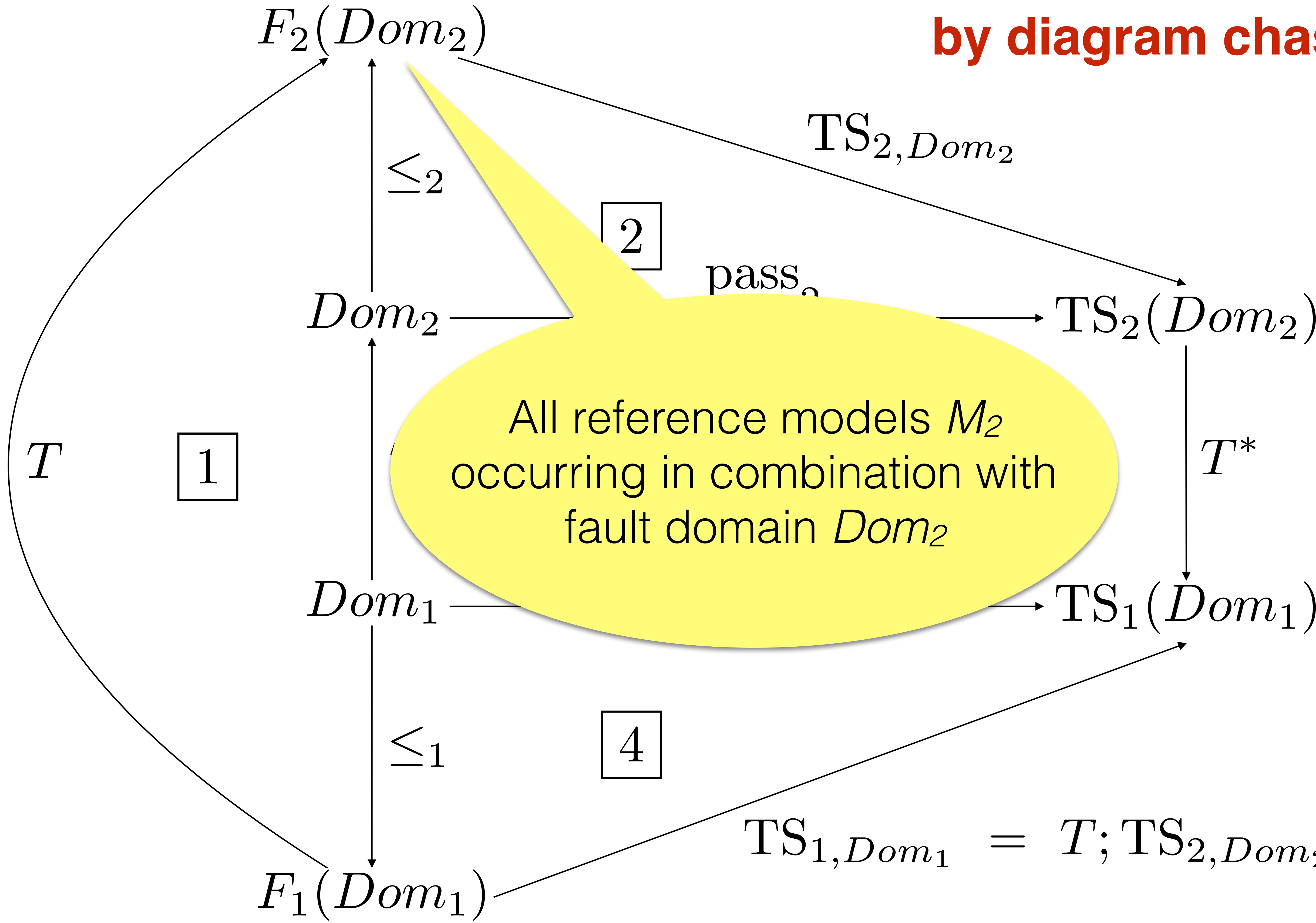
Wen-ling Huang, Jan Peleska:
Complete Model-Based Equivalence Class Testing for Nondeterministic Systems.
Submitted to Formal Aspects of Computing, 2015

Proof of Theorem 1 by diagram chasing

$$
\begin{tikzcd}
F_2(Dom_2) \arrow[dr, "\mathrm{TS}_{2,Dom_2}"] & {} \\
Dom_2 \arrow[u, "\leq_2"'] \arrow[r, "\underline{\mathrm{pass}}_2"] \arrow[ur, phantom, "\boxed{2}"] \arrow[dr, phantom, "\boxed{3}"] & \mathrm{TS}_2(Dom_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"'] \arrow[r, "\underline{\mathrm{pass}}_1"] \arrow[d, "\leq_1"] \arrow[dr, phantom, "\boxed{4}"] & \mathrm{TS}_1(Dom_1) \\
F_1(Dom_1) \arrow[uuu, bend left=60, "T"] \arrow[uuu, phantom, bend left=30, "\boxed{1}"] \arrow[ur] & {}
\end{tikzcd}
$$

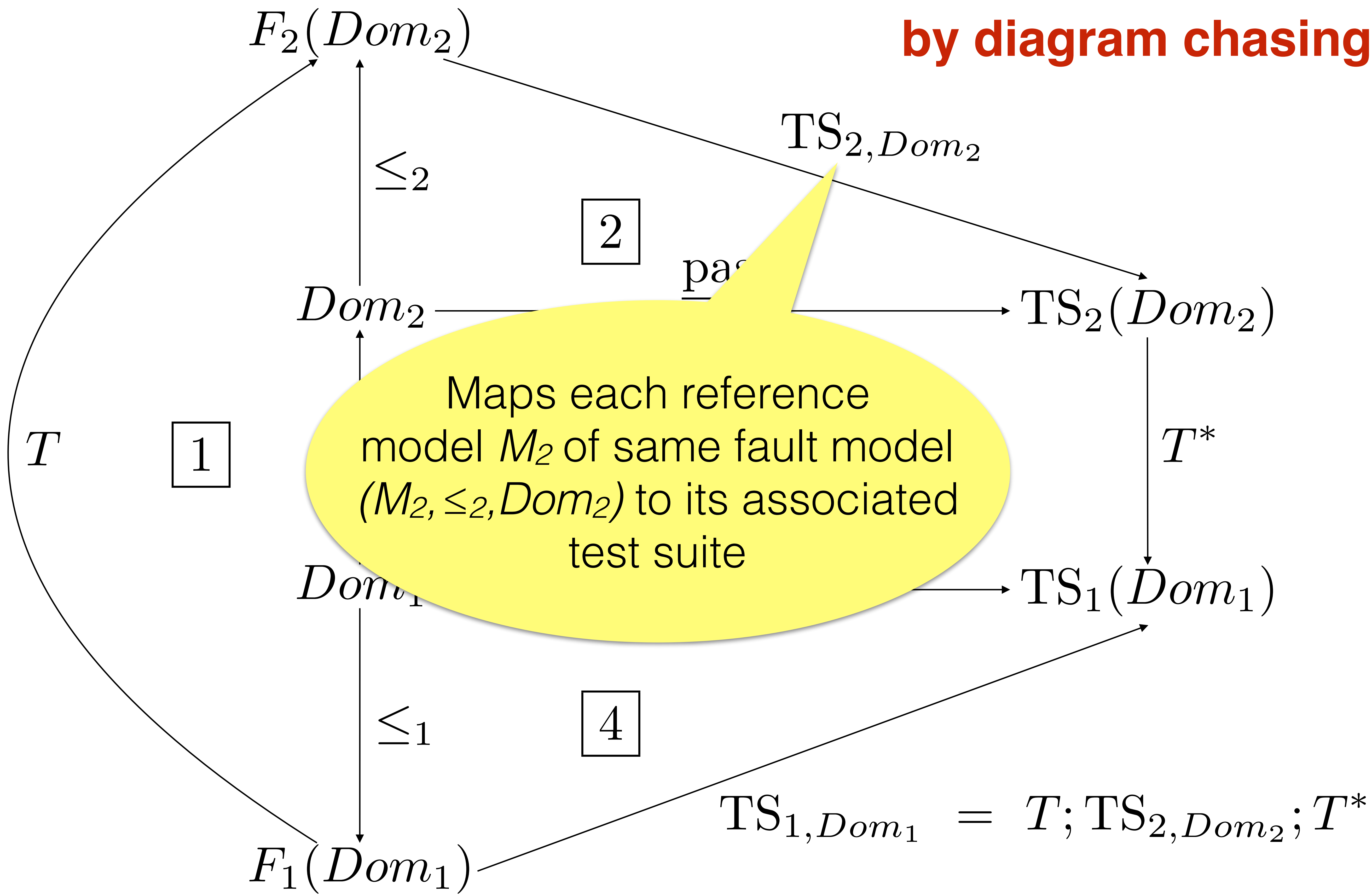
$$\mathrm{TS}_{1,Dom_1} \;=\; T; \mathrm{TS}_{2,Dom_2}; T^*$$

Proof of Theorem 1 by diagram chasing

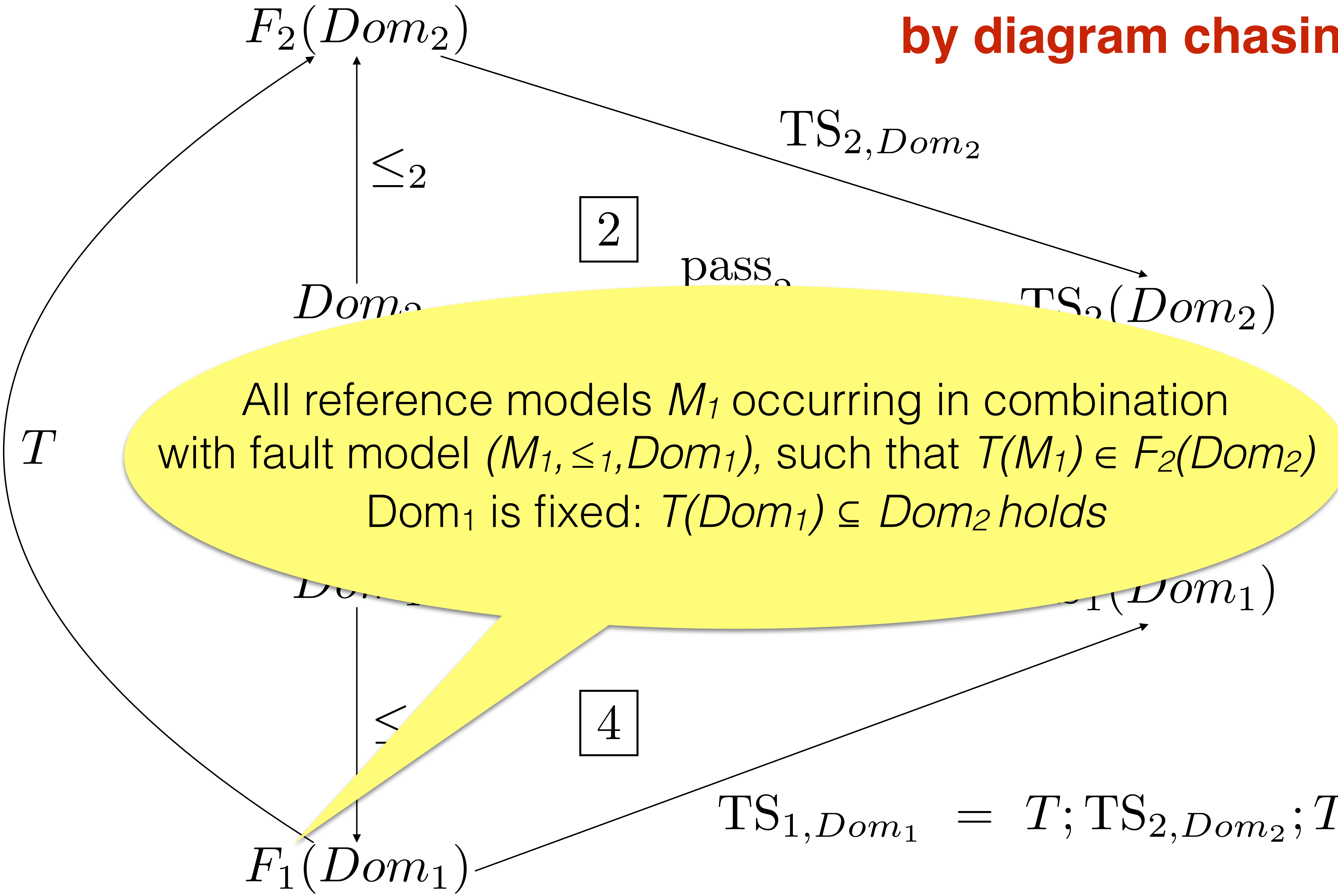


$$\mathrm{TS}_{1,Dom_1} \;=\; T;\mathrm{TS}_{2,Dom_2};T^*$$

Proof of Theorem 1 by diagram chasing

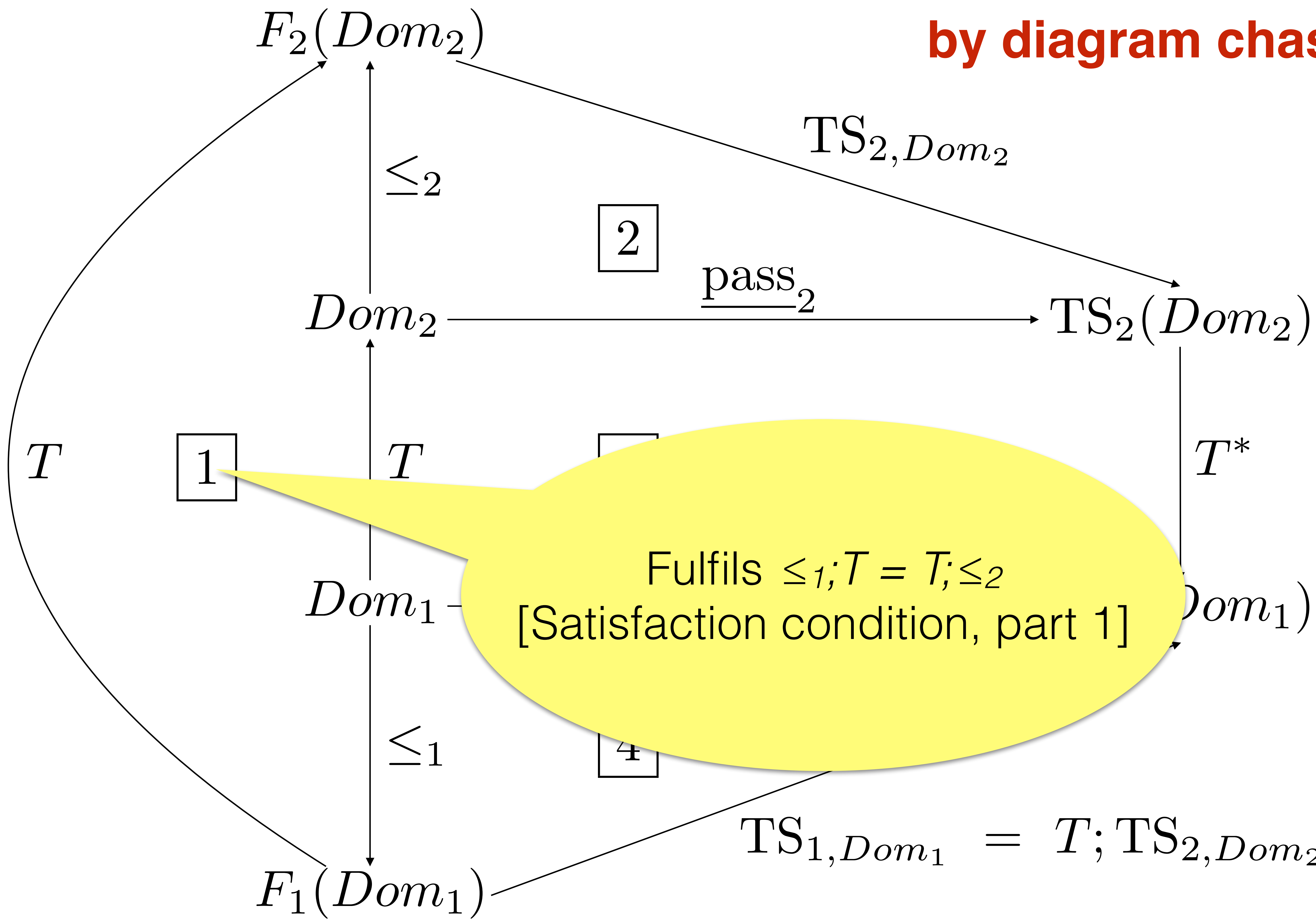


Proof of Theorem 1 by diagram chasing



$$\mathrm{TS}_{1,Dom_1} \;=\; T;\mathrm{TS}_{2,Dom_2};T^*$$

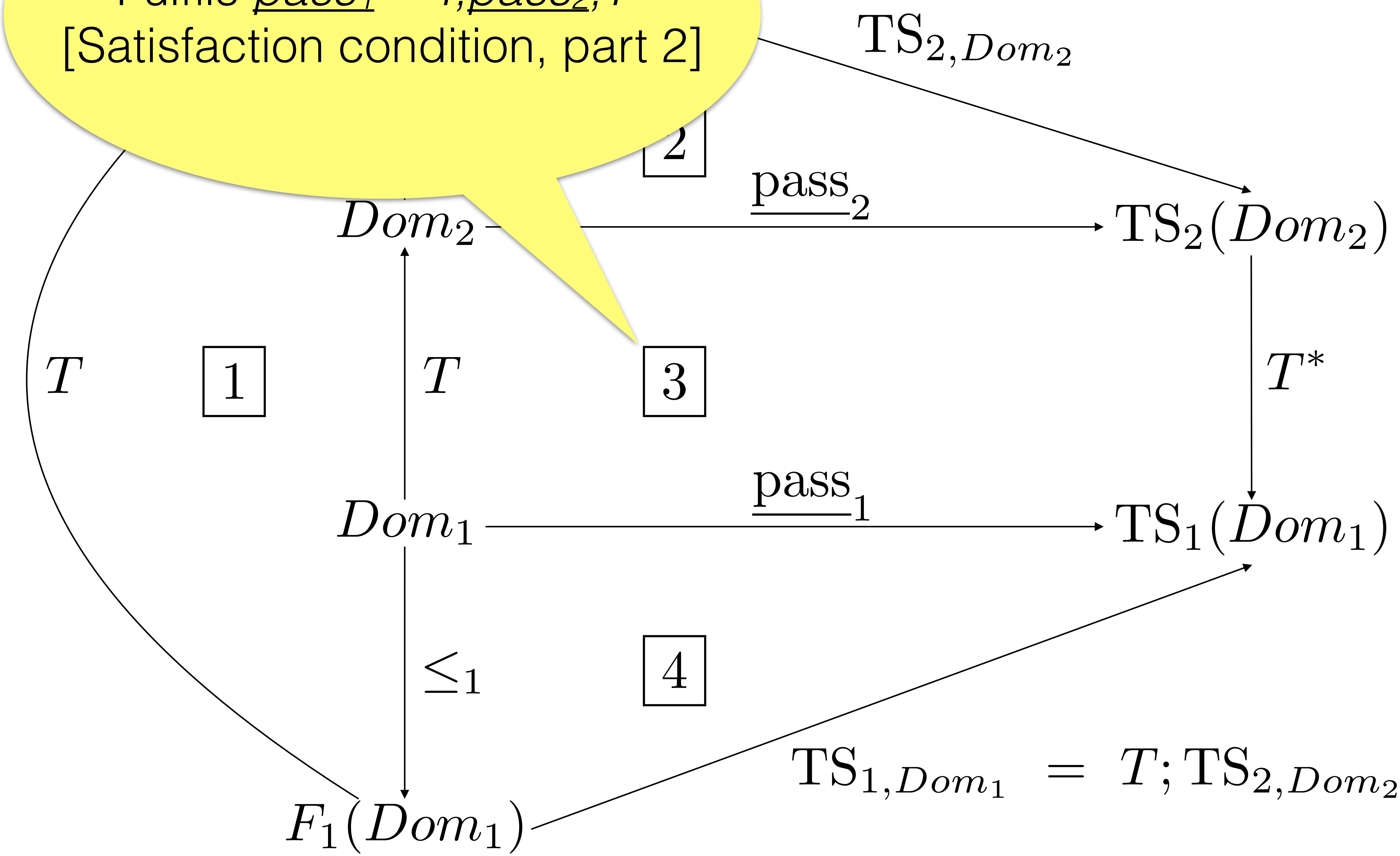
Proof of Theorem 1 by diagram chasing



$$\mathrm{TS}_{1,Dom_1} \;=\; T;\mathrm{TS}_{2,Dom_2};T^*$$

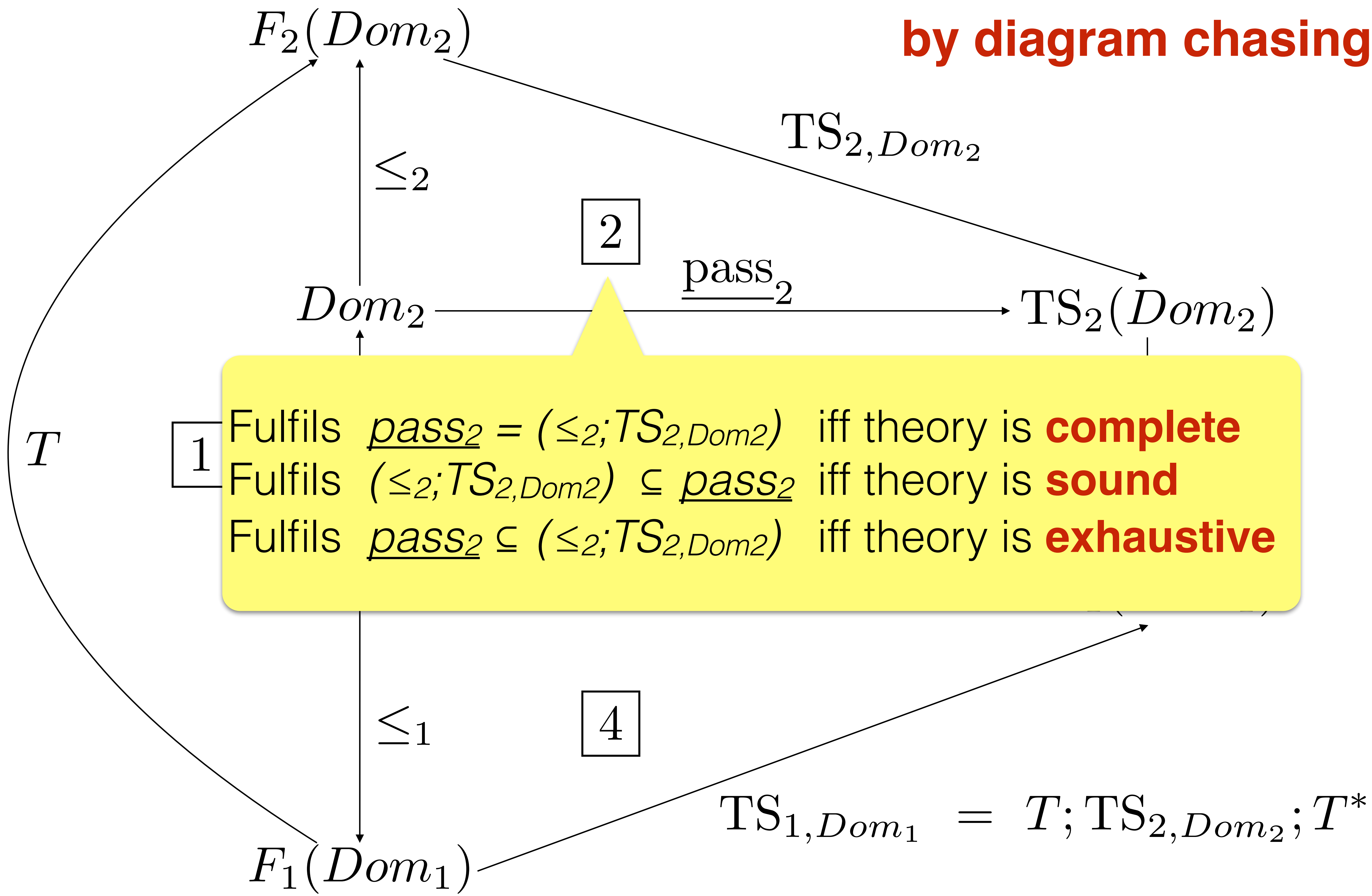
Proof of Theorem 1 by diagram chasing

Fulfils *pass₁ = T;pass₂;T** [Satisfaction condition, part 2]

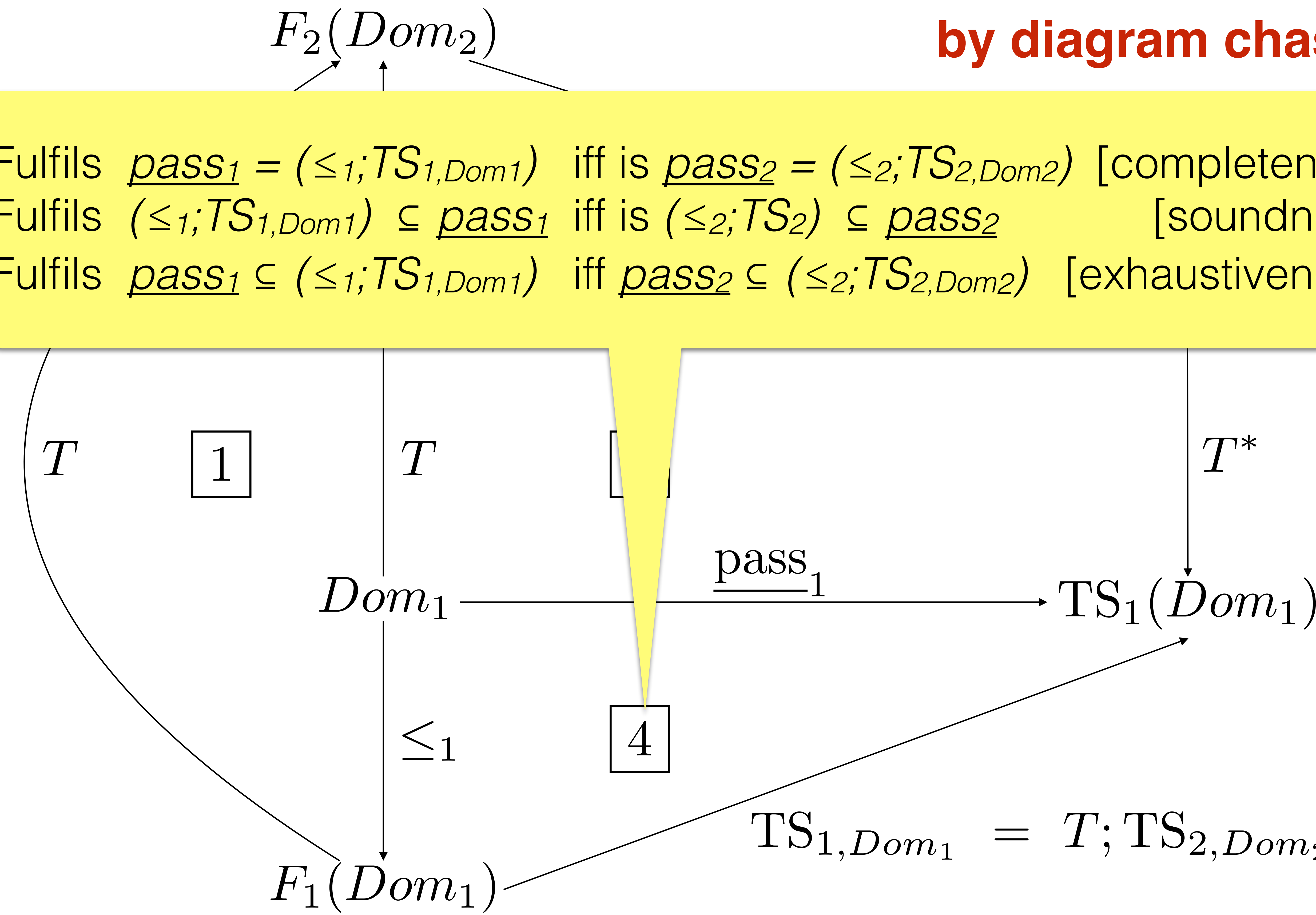


$$\mathrm{TS}_{1,Dom_1} \;=\; T;\mathrm{TS}_{2,Dom_2};T^*$$

Proof of Theorem 1 by diagram chasing



Proof of Theorem 1 by diagram chasing



Fulfils $\underline{pass_1} = (\leq_1;TS_{1,Dom1})$ iff is $\underline{pass_2} = (\leq_2;TS_{2,Dom2})$ [completeness]

Fulfils $(\leq_1;TS_{1,Dom1}) \subseteq \underline{pass_1}$ iff is $(\leq_2;TS_2) \subseteq \underline{pass_2}$ [soundness]

Fulfils $\underline{pass_1} \subseteq (\leq_1;TS_{1,Dom1})$ iff $\underline{pass_2} \subseteq (\leq_2;TS_{2,Dom2})$ [exhaustiveness]

$$TS_{1,Dom_1} = T; TS_{2,Dom_2}; T^*$$

Proof of Theorem 1 by diagram chasing

$$
\begin{tikzcd}[column sep=huge]
F_2(Dom_2) \arrow[dr, "\mathrm{TS}_2"] & \\
Dom_2 \arrow[u, "\leq_2"'] \arrow[r, "\underline{\mathrm{pass}}_2"] \arrow[ur, phantom, "\boxed{2}"] & \mathrm{TS}_2(Dom_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"'] \arrow[d, "\leq_1"'] \arrow[r, green, thick, "\underline{\mathrm{pass}}_1"] \arrow[ur, phantom, "\boxed{3}"] & \mathrm{TS}_1(Dom_1) \\
F_1(Dom_1) \arrow[ur] \arrow[uuu, bend left=60, "T", "\boxed{1}"'] \arrow[r, phantom, "\boxed{4}"] &
\end{tikzcd}
$$

$$\mathrm{TS}_1 = T; \mathrm{TS}_2; T^*$$

Proof of Theorem 1 by diagram chasing

$$
\begin{tikzcd}
F_2(Dom_2) \arrow[dr, "\mathrm{TS}_2"] \arrow[dr, phantom, "\boxed{2}"'] & \\
Dom_2 \arrow[u, "\leq_2"'] \arrow[r, "\underline{\mathrm{pass}}_2"] \arrow[dr, phantom, "\boxed{3}"] & \mathrm{TS}_2(Dom_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"'] \arrow[r, "\underline{\mathrm{pass}}_1"] \arrow[d, "\leq_1"] \arrow[dr, phantom, "\boxed{4}"] & \mathrm{TS}_1(Dom_1) \\
F_1(Dom_1) \arrow[ur, "\mathrm{TS}_1 = T; \mathrm{TS}_2; T^*"'] \arrow[uuu, bend left=60, "T"] \arrow[uuu, phantom, bend left=30, "\boxed{1}"] &
\end{tikzcd}
$$

Proof of Theorem 1 by diagram chasing

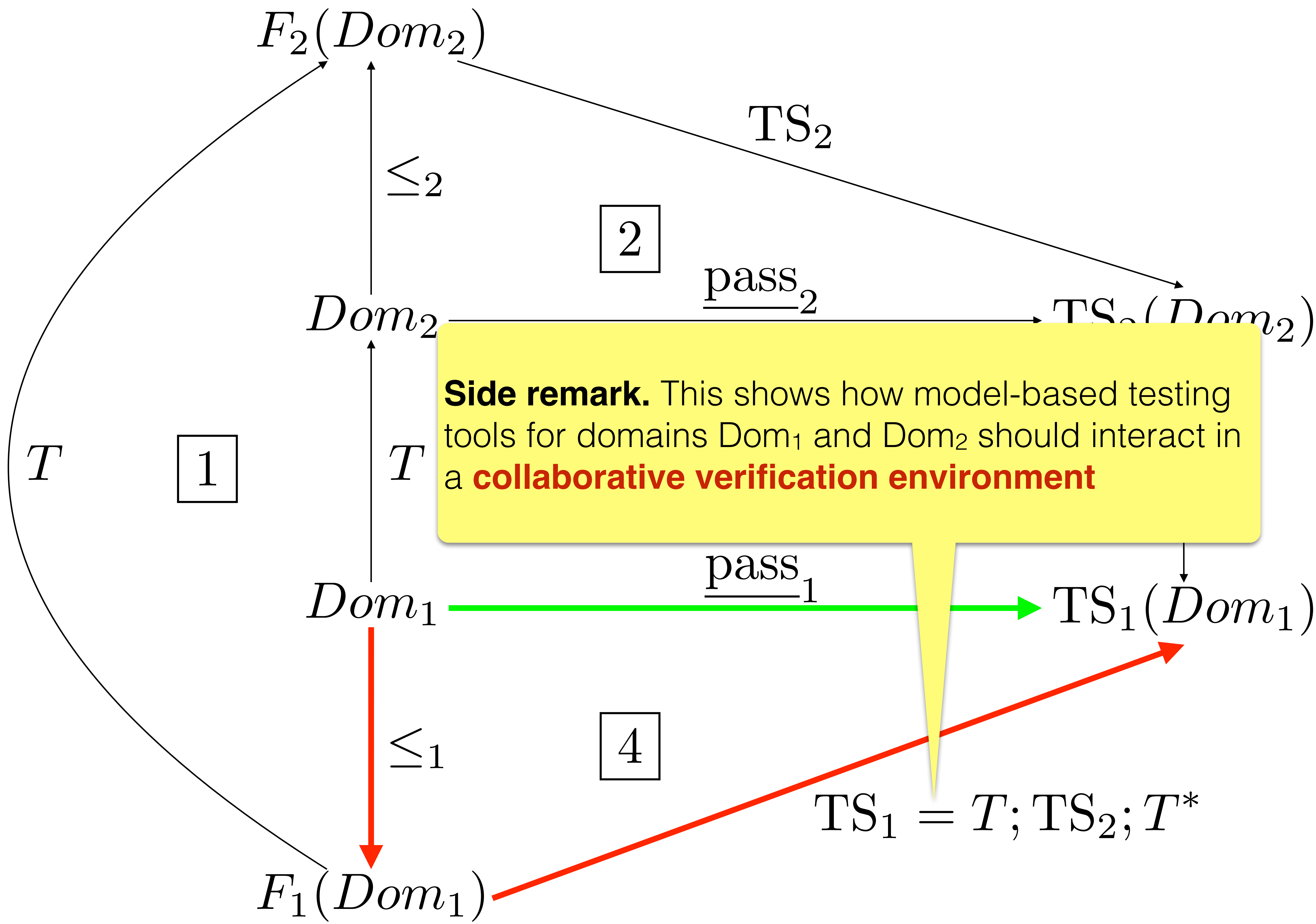
$$
\begin{tikzcd}[column sep=huge, row sep=large]
F_2(Dom_2) \arrow[dr, "\mathrm{TS}_2"] & \\
Dom_2 \arrow[u, "\leq_2"] \arrow[r, "\underline{\mathrm{pass}}_2"] \arrow[ur, phantom, "\boxed{2}"] \arrow[dr, phantom, "\boxed{3}"] & \mathrm{TS}_2(Dom_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"] \arrow[r, "\underline{\mathrm{pass}}_1"] \arrow[d, "\leq_1"] \arrow[dr, phantom, "\boxed{4}"] & \mathrm{TS}_1(Dom_1) \\
F_1(Dom_1) \arrow[ur, swap, "\mathrm{TS}_1 = T; \mathrm{TS}_2; T^*"] \arrow[uuu, bend left=60, "T"] \arrow[uuu, phantom, "\boxed{1}"] &
\end{tikzcd}
$$

Proof of Theorem 1 by diagram chasing

$$
\begin{tikzcd}[column sep=huge, row sep=large]
F_2(Dom_2) \arrow[dr, "\mathrm{TS}_2"] & \\
Dom_2 \arrow[u, "\leq_2"'] \arrow[r, "\underline{\mathrm{pass}}_2"] \arrow[ur, phantom, "\boxed{2}"] \arrow[dr, phantom, "\boxed{3}"] & \mathrm{TS}_2(Dom_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"'] \arrow[r, "\underline{\mathrm{pass}}_1"] \arrow[d, "\leq_1"] \arrow[dr, phantom, "\boxed{4}"] & \mathrm{TS}_1(Dom_1) \\
F_1(Dom_1) \arrow[ur, "\mathrm{TS}_1 = T; \mathrm{TS}_2; T^*"'] \arrow[uuu, bend left=60, "T", "\boxed{1}"'] &
\end{tikzcd}
$$

Proof of Theorem 1 by diagram chasing

$$
\begin{tikzcd}
F_2(Dom_2) \arrow[drr, "\mathrm{TS}_2"] \arrow[dr, phantom, "\boxed{2}"] & & \\
Dom_2 \arrow[u, "\leq_2"'] \arrow[rr, "\underline{\mathrm{pass}}_2"] \arrow[dr, phantom, "\boxed{3}"] \arrow[d, phantom, "\boxed{1}" {left, xshift=-6ex}] & & \mathrm{TS}_2(Dom_2) \arrow[d, "T^*"] \\
Dom_1 \arrow[u, "T"'] \arrow[rr, "\underline{\mathrm{pass}}_1"] \arrow[d, "\leq_1"] \arrow[dr, phantom, "\boxed{4}"] & & \mathrm{TS}_1(Dom_1) \\
F_1(Dom_1) \arrow[uuu, bend left=60, "T"] \arrow[urr, "\mathrm{TS}_1 = T; \mathrm{TS}_2; T^*"'] & &
\end{tikzcd}
$$



Side remark. This shows how model-based testing tools for domains Dom_1 and Dom_2 should interact in a **collaborative verification environment**

TTT application

- **Theorem 2.** Every complete (sound, exhaustive) FSM testing theory for

  - language equivalence or

  - language containment

  induces a complete (sound, exhaustive) **equivalence class partition testing theory** with analogous conformance relations for Kripke structures with **infinite input domains**, bounded nondeterminism, and finite internal state and finite outputs

Wen-ling Huang, Jan Peleska:
Complete Model-Based Equivalence Class Testing.
Int J Softw Tools Techno Transfer,
DOI 10.1007/s10009-014-0356-8., 2014

Wen-ling Huang, Jan Peleska:
Complete Model-Based Equivalence Class Testing for Nondeterministic Systems.
Submitted to Formal Aspects of Computing, 2015

TTT-application

- **Step 1. Transform the transition relation**
  - Create a transition relation of the model
  - Separate input variables, internal model variables, and output variables, by enumerating the latter
  - Aggregate sequences of transitions between transient states into a single transition leading to a quiescent post-state

TTT-application

- Step 1. Transform the transition relation
  - This leads to transition relation of the form

$$
\begin{aligned}
\mathcal{R} \quad &\equiv \quad \bigvee_{i \in \mathrm{IDX}} \big(\alpha_i \wedge (\boldsymbol{m}, \boldsymbol{y}) = (\boldsymbol{d}_i, \boldsymbol{e}_i) \wedge (\boldsymbol{m}', \boldsymbol{y}') = (\boldsymbol{d}_i, \boldsymbol{e}_i)\big) \\
&\vee \quad \bigvee_{(i,j) \in J} \big(g_{i,j} \wedge (\boldsymbol{m}, \boldsymbol{y}) = (\boldsymbol{d}_i, \boldsymbol{e}_i) \wedge (\boldsymbol{m}', \boldsymbol{y}') = (\boldsymbol{d}_j, \boldsymbol{e}_j)\big)
\end{aligned}
$$

with

- Stability conditions α_i
- Jump conditions $g_{i,j}$
- Only input variables occur free in $\alpha_i, g_{i,j}$

TTT-application

- **Step 2. Calculation of input equivalence classes**
  - Each satisfiable solution of

$$\Phi_f \equiv \bigwedge_{i \in \mathrm{IDX}} g_{i,f(i)} \text{ with } f : \mathrm{IDX} \to \mathrm{IDX} \text{ permutation}$$

  specifies one input equivalence class

TTT-application

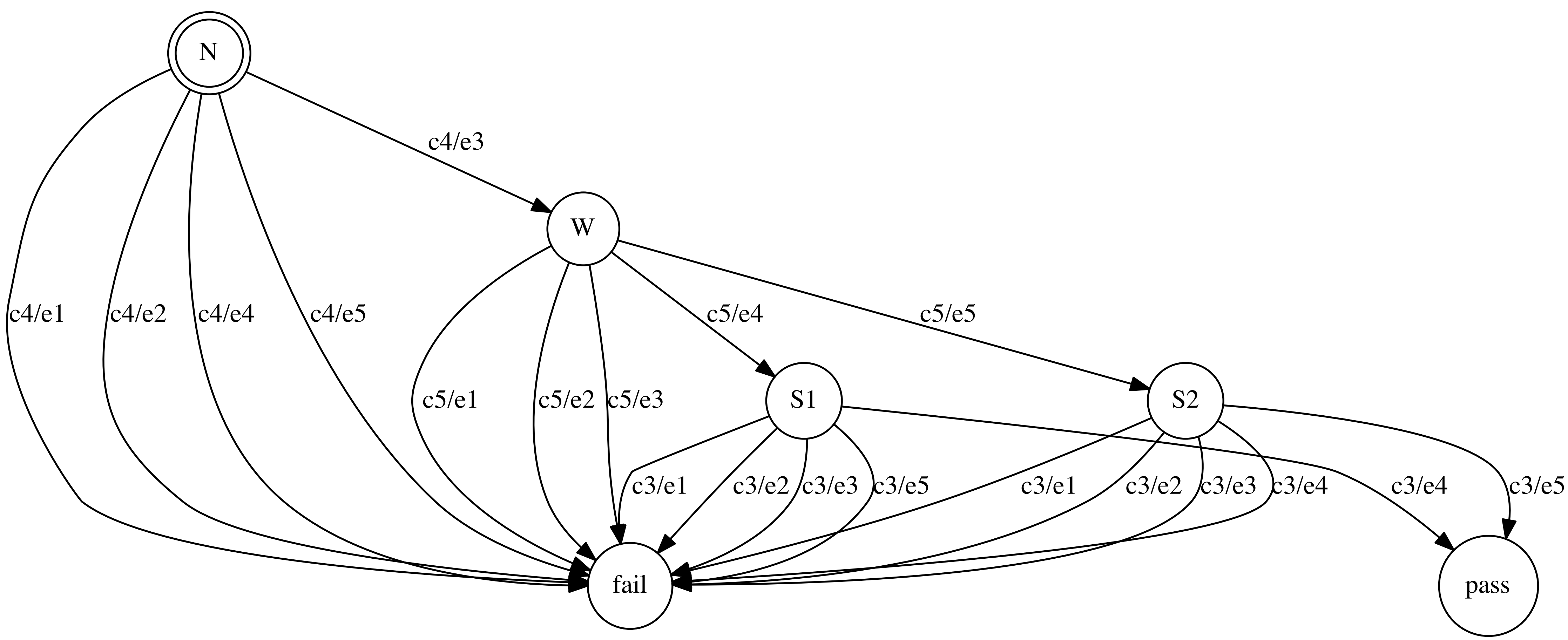
- **Step 3. Creation of the model map**
  - Map Kripke model to minimal, observable FSM with

$$
\begin{aligned}
\text{Input alphabet} \quad \Sigma_I &= \quad \{\Phi_f \mid \Phi_f \text{ is feasible}\} \\
\text{Output alphabet} \quad \Sigma_O &= \quad \text{finite output domain of Kripke model} \\
\text{Internal states} \quad Q &= \quad \{q_i \mid i \in \mathrm{IDX}\} \\
\text{Transition relation} \quad h &= \quad \{(q_i, \Phi_f, \boldsymbol{e}_j, q_j) \mid f(i) = j\}
\end{aligned}
$$

TTT-application

- **Step 4. Creation of the test case map**
  - FSM test cases are acyclic, terminating, single-input, output-complete FSMs
  - FSM test cases interact with the FSM to be tested via language intersection as „parallel operator“
  - FSM test inputs state-dependent value to SUT
  - FSM test accepts SUT output and transits into next state with new input or into fail state

FSM test case

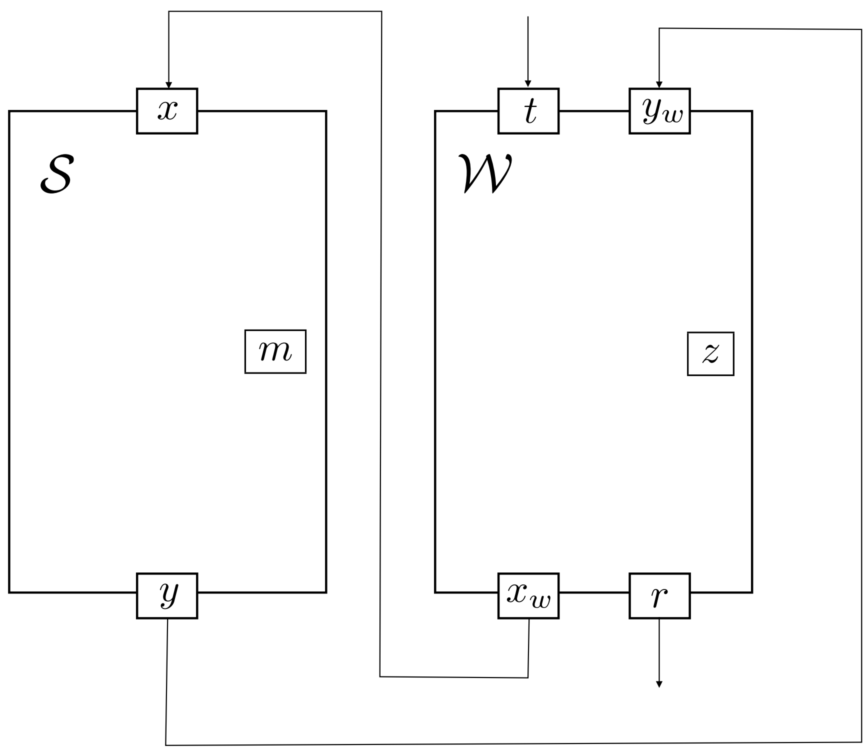


TTT-application

- **Step 4. Creation of the test case map**

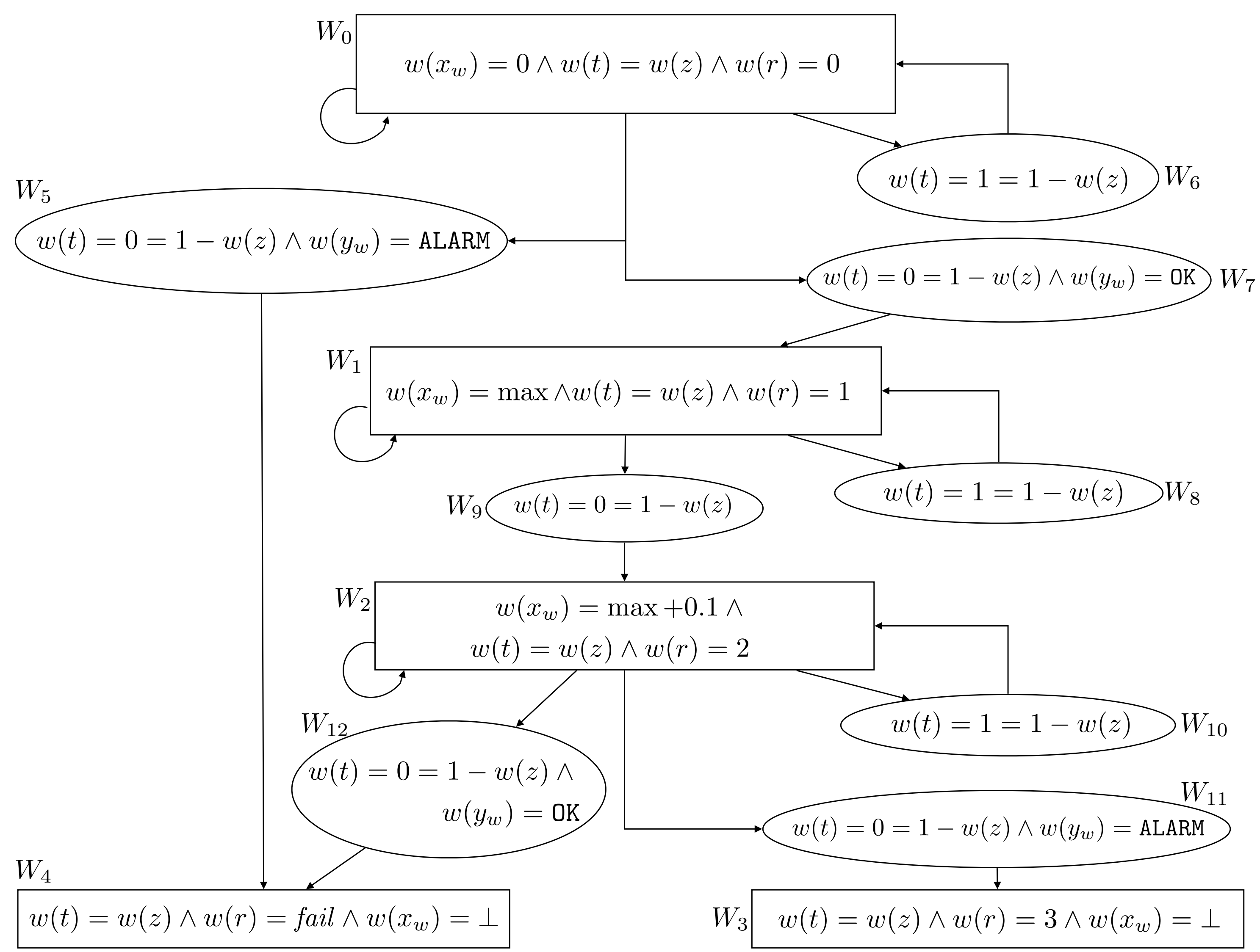
    - Kripke structure test cases interact (this is one option) synchronously with the SUT

    - In contrast to FSM test cases, inputs to the SUT are strictly separated from monitoring of outputs



TTT-application

- **Step 4. Creation of the test case map**
  - Consequently, one FSM test step leads to a more complex Kripke test step involving several transitions



TTT-application

- **Step 5. Proof of the satisfaction condition**
  - The proof is independent on the selection of representatives from each equivalence class, whenever this class occurs as an input in an FSM test case
  - Consequently, the test strategy for Kripke structures can be combined with random selection of input data from each class

Theory translation – model-theoretic underpinning

- **Alternative A. Theory of Institutions**

Test case map above corresponds to **sentence translation map** in theory of institutions – Need **Grothendiek Institutions**

Razvan Diaconescu: **Institution-independent Model-Theory.** Birkhäuser Verlag, Basel, Boston, Berlin, 2008

Joseph A. Goguen, Rod M. Burstall: **Institutions: Abstract Model Theory for Specification and Programming.** J. ACM 39(1): 95-146 (1992)

- Semantics for CPS – time for a change of paradigm?
- **Multiple formalisms in CPS modelling**
  - Example 1. Testing theories and collaborative tool environments
  - **Example 2. Verification of emergent properties**
- Conclusions and future work

Multiple formalisms in CPS modelling – Example 2. Verification of emergent properties

Recall – train onboard speed control

Train engine driver brake command

man_on, man_off

Onboard main controller

PLC brake controller

Automated braking command

auto_on, auto_off

trigger, release

Current speed

Current maximum speed

Emergency brake

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**

Verification of emergent properties

- **Observations**
  - FSM I/O-events x/y can be mapped to CSP channel events x.y
  - FSM parallel composition by intersection is similar to synchronous channel communication of CSP processes
  - CSP failures models can be represented by normalised transition graphs

A.W. Roscoe:
Model-Checking CSP. In **A Classical Mind: Essays in Honour of C.A.R. Hoare.**
Prentice Hall International (UK), 1994

Alternative approach

- **Alternative B. Approach based on Unifying Theories of Programming UTP**
  - „Programs are predicates“ – no distinction between models and sentences
  - Theories are made up from alphabets, signatures, and healthiness conditions
  - Conformance is expressed by implication $[P \Rightarrow Q]$ („P refines Q“)
  - Model, sentence, and theory translation is enabled by the existence of Galois connections

Jifeng He, C. A. R. Hoare: Unifying theories of programming. RelMiCS 1998: 97-99

Verification of emergent properties

- **Procedure**
  - Create UTP theories for
    - ◆ Sub-class of Kripke structures (sequential nondeterministic programs) with LTL safety formulas for property specifications,
    - ◆ FSMs with property specification by testing automata
    - ◆ CSP failures model with failures (= trace/refusal) specifications
      P ***sat*** *S(tr,ref)*
    - ◆ CSP transition graphs with CSP-like specifications
      G ***sat*** *S(tr,ref)*

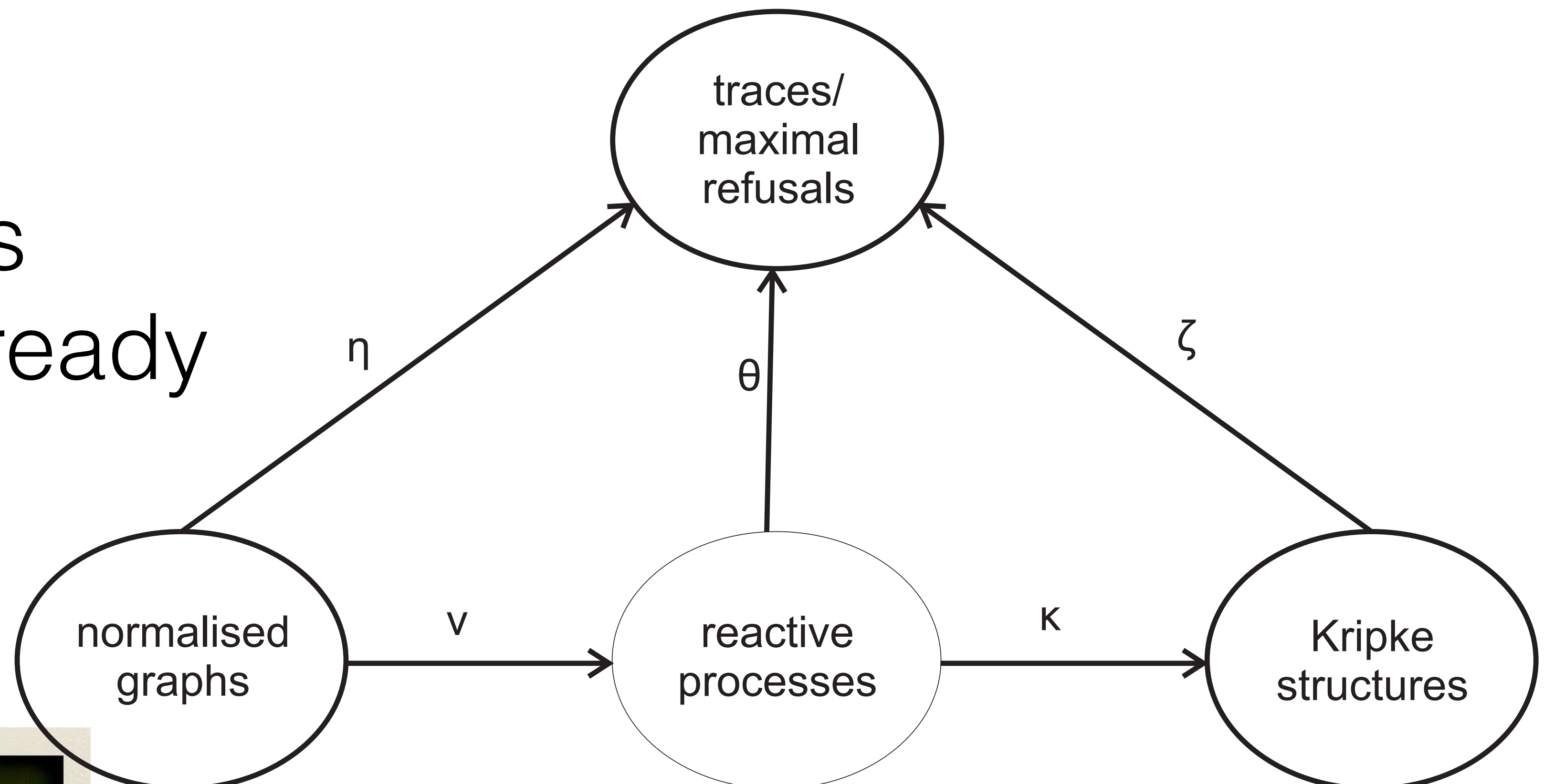
Verification of emergent properties

- **Procedure**
  - Create Galois connections
    - CSP failures models ⇄ CSP transition graphs
    - Sequential nondeterministic programs ⇄ CSP transition graphs
    - FSMs ⇄ CSP transition graphs
  - This allows us to
    - lift the local properties of FSM and Kripke structure to local CSP assertions
    - deduce the required satisfaction relation on CSP level by means of compositional reasoning

Theory translation – model-theoretic underpinning

Application example

- Some of these Galois connections have already been established



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

Conclusions and future work

Conclusions

- We have identified characteristics of CPS challenging the existing semantic approaches to concurrent systems

- Potential solutions to the problems of

  - theory translation

  - verification of emergent properties in presence of multiple formalisms

  have been proposed

Future work

- Evolution of asserted behaviour
  - Inspiration from AI. **Belief systems** and belief revision – CPS components should act optimally in relation to the current status of belief – belief revision should only be necessary within specified boundaries
- Semantic navigation
  - A network of semantics offering different degrees of abstraction
  - Network nodes are connected by **theory translation mappings** – Galois Connections?
- Dynamic re-configuration
  - Simpler methods are available for **bounded-length model investigation**, as used in bounded model checking and model-based testing

Acknowledgements

I would like to express my gratitude to this audience, its organisers, and to my friends and collaborators who inspired and contributed to the ideas presented in this talk.

Ana Cavalcanti, Anne E. Haxthausen, Wen-ling Huang, Christoph Hilken, Felix Hübner, John Fitzgerald, Peter Gorm Larsen, Till Mossakowski, Mohammad Reza Mousavi, Alexandre Petrenko, Markus Roggenbach, Uwe Schulze, Linh Hong Vu, Jim Woodcock, Cornelia Zahlten

The work presented here has been performed in the context of project Implementable Testing Theories for Cyber-physical systems (ITTCPS)
http://www.informatik.uni-bremen.de/agbs/projects/ittcps/index.html