
And now for something completely different

FZI Forschungszentrum Informatik

Contract Automata

A Specification Language for Mode-Based Systems

Alexander Weigl, Joshua Bachmeier, Bernhard Beckert and Mattias Ulbrich

2023-08-09 @ KeY-Symposium ’23

Motivation

State of the Art

Design-By-Contract

Motivation

State of the Art

Design-By-Contract

- Useful for single operation, difficult for *reactive system*
- Need way to express state $\rightarrow$ LTL etc. too fine-grained
- Applicable contract depends only on portion of state (mode)

Motivation

State of the Art

Design-By-Contract

- Useful for single operation, difficult for *reactive system*
- Need way to express state $\rightarrow$ LTL etc. too fine-grained
- Applicable contract depends only on portion of state (mode)

Motivation

State of the Art

Design-By-Contract

- Useful for single operation, difficult for *reactive system*
- Need way to express state $\rightarrow$ LTL etc. too fine-grained
- Applicable contract depends only on portion of state (mode)

Solution

Contract Automata: Contract-based specification with FSM

Motivation

State of the Art

Design-By-Contract

- Useful for single operation, difficult for *reactive system*
- Need way to express state $\rightarrow$ LTL etc. too fine-grained
- Applicable contract depends only on portion of state (mode)

Solution

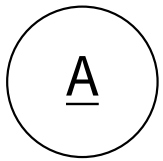
Contract Automata: Contract-based specification with FSM

Focus of this presentation

Specification mechanism, Concept, High-Level

Contract Automata

- Like nondet. FSM but every edge has *assumption*/*guarantee*
- Mode may represent internal system or environment state

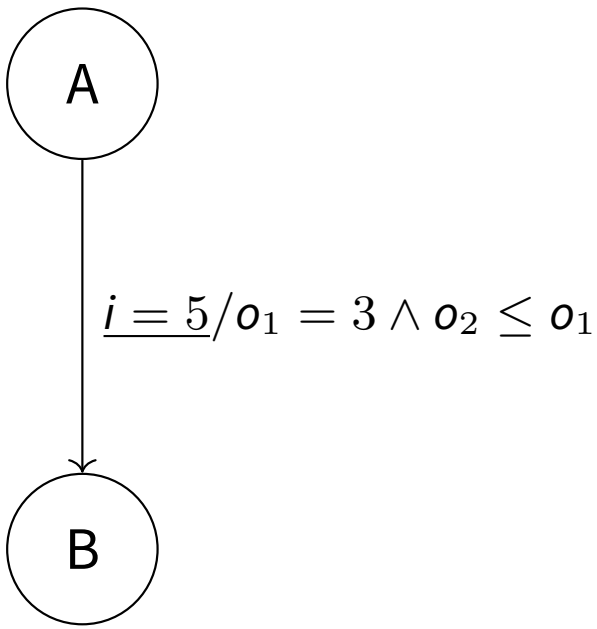


Contract Automata

- Like nondet. FSM but every edge has *assumption*/*guarantee*
- Mode may represent internal system or environment state

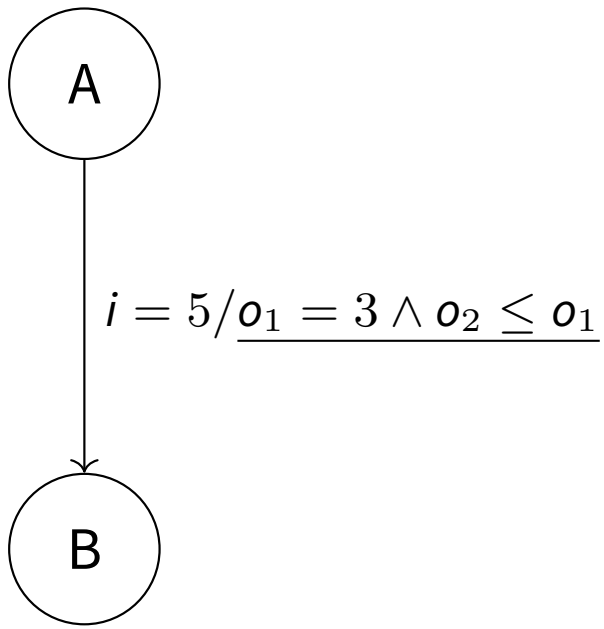
Contract Automata

- Like nondet. FSM but every edge has *assumption*/*guarantee*
- Mode may represent internal system or environment state



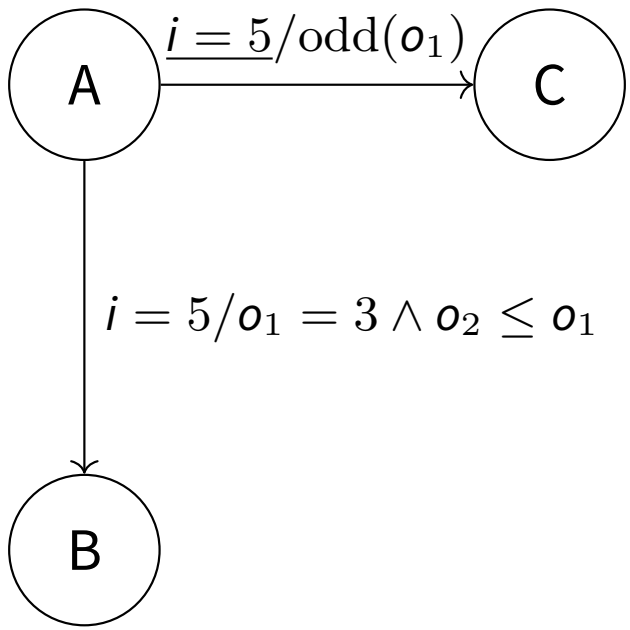
Contract Automata

- Like nondet. FSM but every edge has *assumption*/*guarantee*
- Mode may represent internal system or environment state



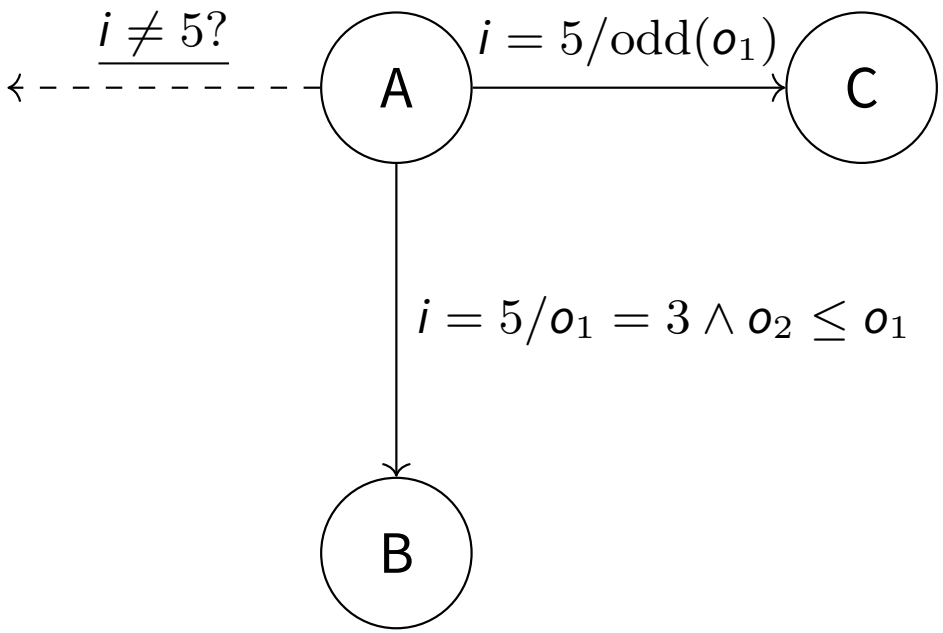
Contract Automata

- Like nondet. FSM but every edge has *assumption*/*guarantee*
- Mode may represent internal system or environment state



Contract Automata

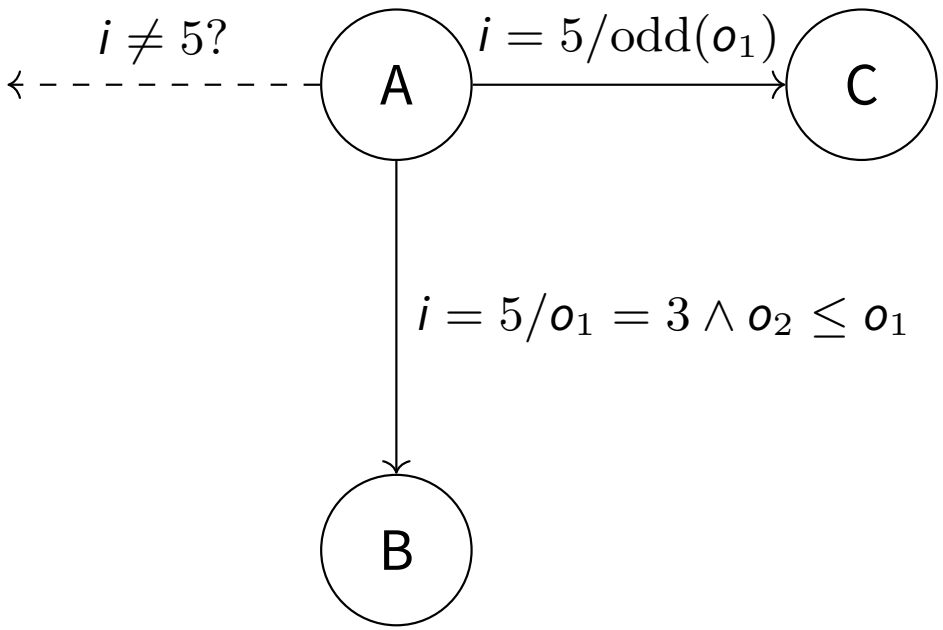
- Like nondet. FSM but every edge has *assumption*/*guarantee*
- Mode may represent internal system or environment state



- "Uncovered" (i.e. valid): if no assumption holds

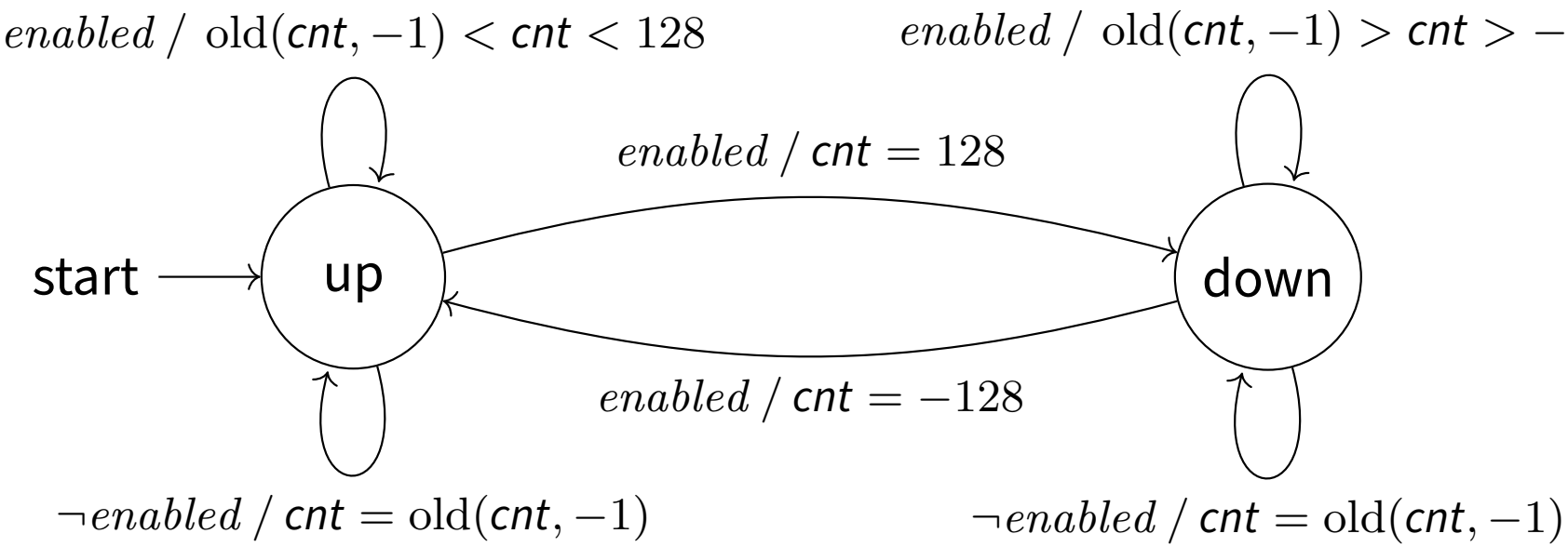
Contract Automata

- Like nondet. FSM but every edge has *assumption*/*guarantee*
- Mode may represent internal system or environment state



- "Uncovered" (i.e. valid): if no assumption holds
- Violation: if any assumption but no guarantee holds

Contract Automata



Reactive Systems

Response Function

$$R : \mathcal{I}^* \to \mathcal{O}$$

- Map inputs *up to now* to next output

Reactive Systems

Response Function

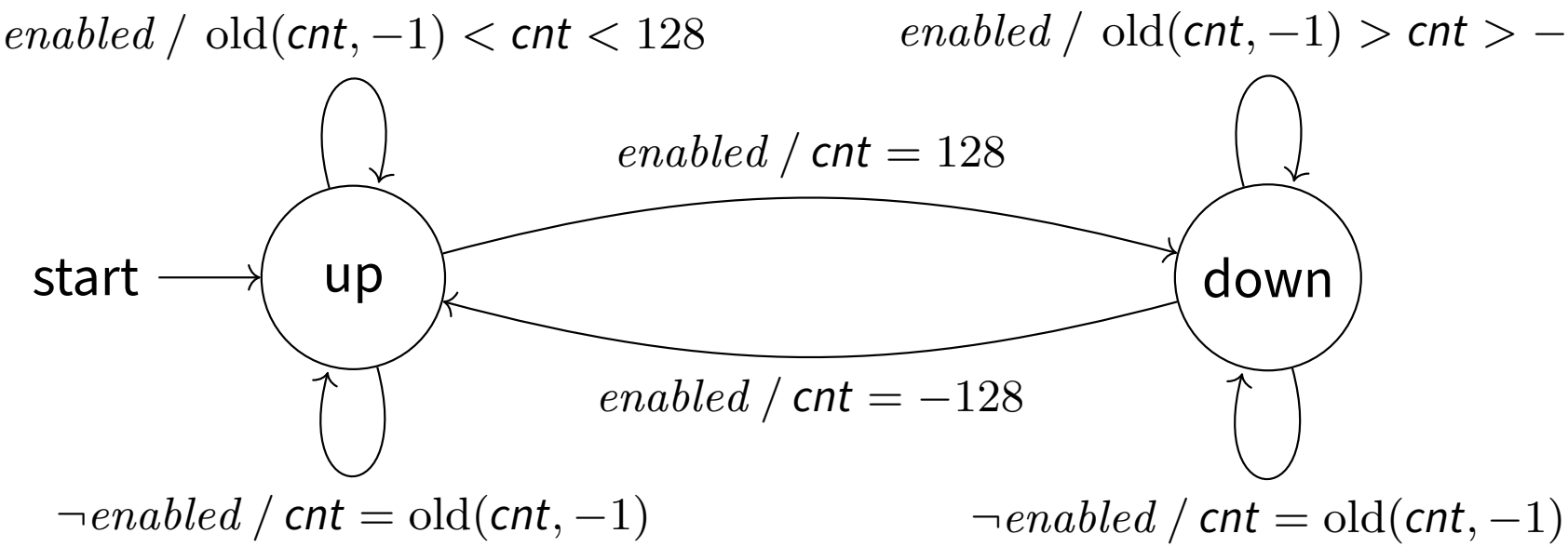
$$R : \mathcal{I}^* \to \mathcal{O}$$

System Behaviour

$$\mathcal{B}(R) = \{((i_0, o_0), (i_1, o_1), \dots) | R(i_0, \dots, i_n) = o_n, n \in \mathbb{N}\}$$

- Map inputs *up to now* to next output
- Behaviour is set of infinite I/O-traces

Possible Behavior of Counter



Traces

$$\mathcal{B}(R) = \{((t,1),(f,1),(t,2),\dots),((t,1),(t,2),(t,3),\dots),\dots\}$$

but also

$$\mathcal{B}(R) = \{((t,5),(t,42),\dots),\dots\},$$
$$\mathcal{B}(R) = \{((t,64),(t,128),(t,64),(t,32),\dots),\dots\}$$

Semantics of Compliance

Intuitively

- At every step: “Inputs satisfy assumptions $\rightarrow$ Output must satisfy guarantee”
- Systems takes path through CA that satisfies contracts on each edge

Semantics of Compliance

Intuitively

- At every step: “Inputs satisfy assumptions $\rightarrow$ Output must satisfy guarantee”
- Systems takes path through CA that satisfies contracts on each edge

Caveats

- Mode not explicit in system
- Bridge gap between syntactic “contract paths” and semantic I/O-traces

Game Semantics (simplified)

Input: Reactive Systeme $R : \mathcal{I}^* \to \mathcal{O}$ and CA

$\hat{S} \leftarrow S_0$; $\quad \bar{\sigma} \leftarrow \varepsilon$; $\quad \bar{\iota} \leftarrow \varepsilon$;

while *true* **do**

  Environment chooses input $i \in \mathcal{I}$;

  $\bar{\iota} \leftarrow \bar{\iota} \cdot i$;

  System computes response $o = R(\bar{\iota}) \in \mathcal{O}$;

  $\bar{\sigma} \leftarrow \bar{\sigma} \cdot (i, o)$;

  $\hat{E} \leftarrow \{(s, c, s') \in \delta \mid \bar{\sigma} \models \mathrm{assume}(c)\}$;

  if $\hat{E} = \emptyset$ **then**

    return *System wins*;

  end

  $\hat{S} \leftarrow \{s' \mid (s, c, s') \in \hat{E} \wedge \bar{\sigma} \models \mathrm{guarantee}(c)\}$;

  if $\hat{S} = \emptyset$ **then**

    return *Environment wins*;

  end

end

Additional Concepts

Refinement

$A \preceq B$

- Based on LSP: $c^A \preceq c^B$ iff.
  $\mathrm{assume}(c^B) \rightarrow \mathrm{assume}(c^A)$ and
  $\mathrm{guarantee}(c^A) \rightarrow \mathrm{guarantee}(c^B)$
- All contracts along path must be refinements
- Every path in A is refinement of path in B

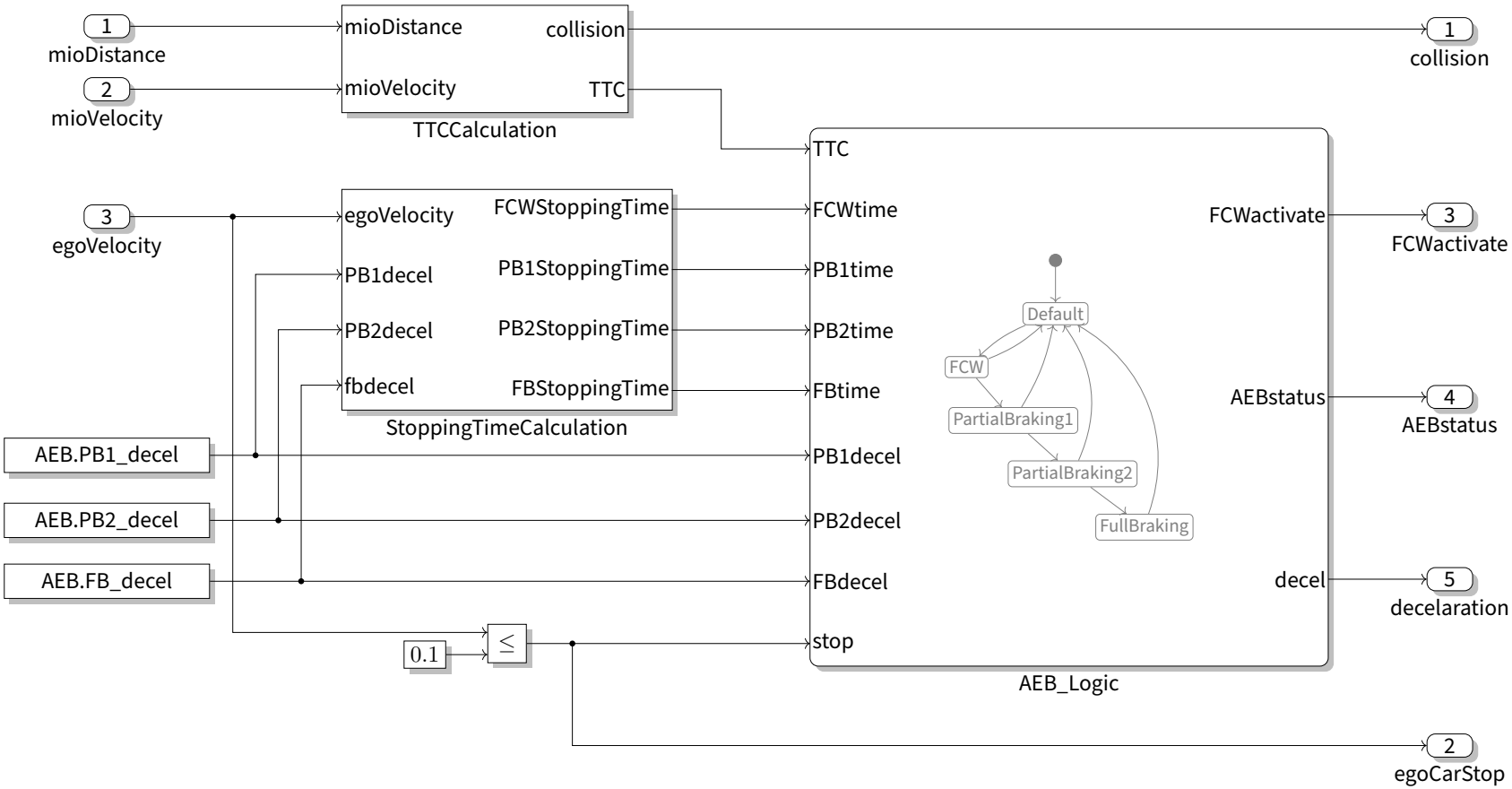
Composition

- Sequential execution: $C := A \overset{oi}{\rightarrowtail} B$
- *oi* defines *output* to *input* mapping

Composition

- Sequential execution: $C := A \overset{oi}{\rightarrowtail} B$
- *oi* defines *output* to *input* mapping
- Parallel execution: $A \parallel B := A \overset{\emptyset}{\rightarrowtail} B$

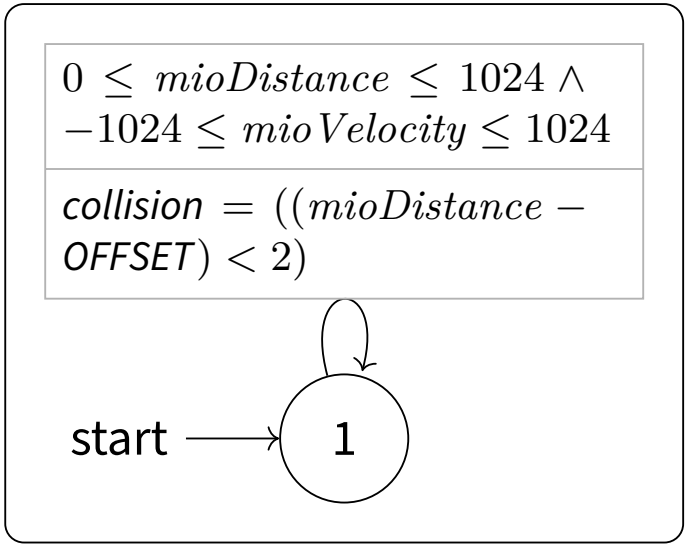
Emergency Break Assistant



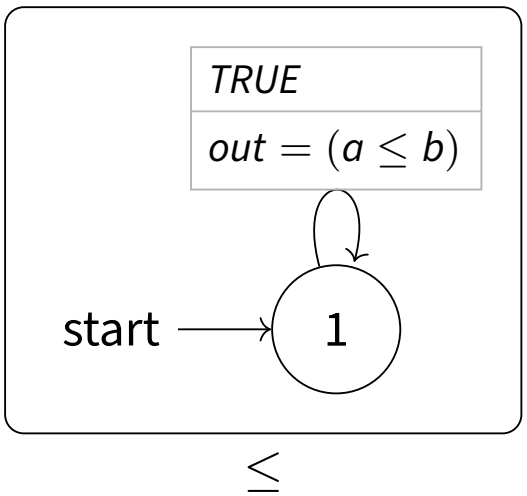
Formula

$$(\text{TTTCalculation} \parallel \text{StoppingTimeCalculation} \parallel \leq) \overset{oi}{\rightarrowtail} \text{AEB\_Logic} \preceq \text{AEB}$$

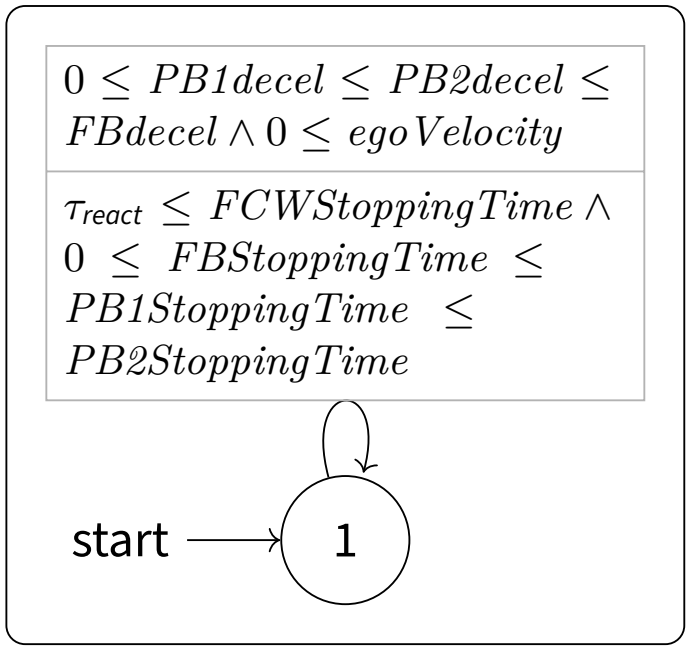
Subsystems I



TTCCalcuation

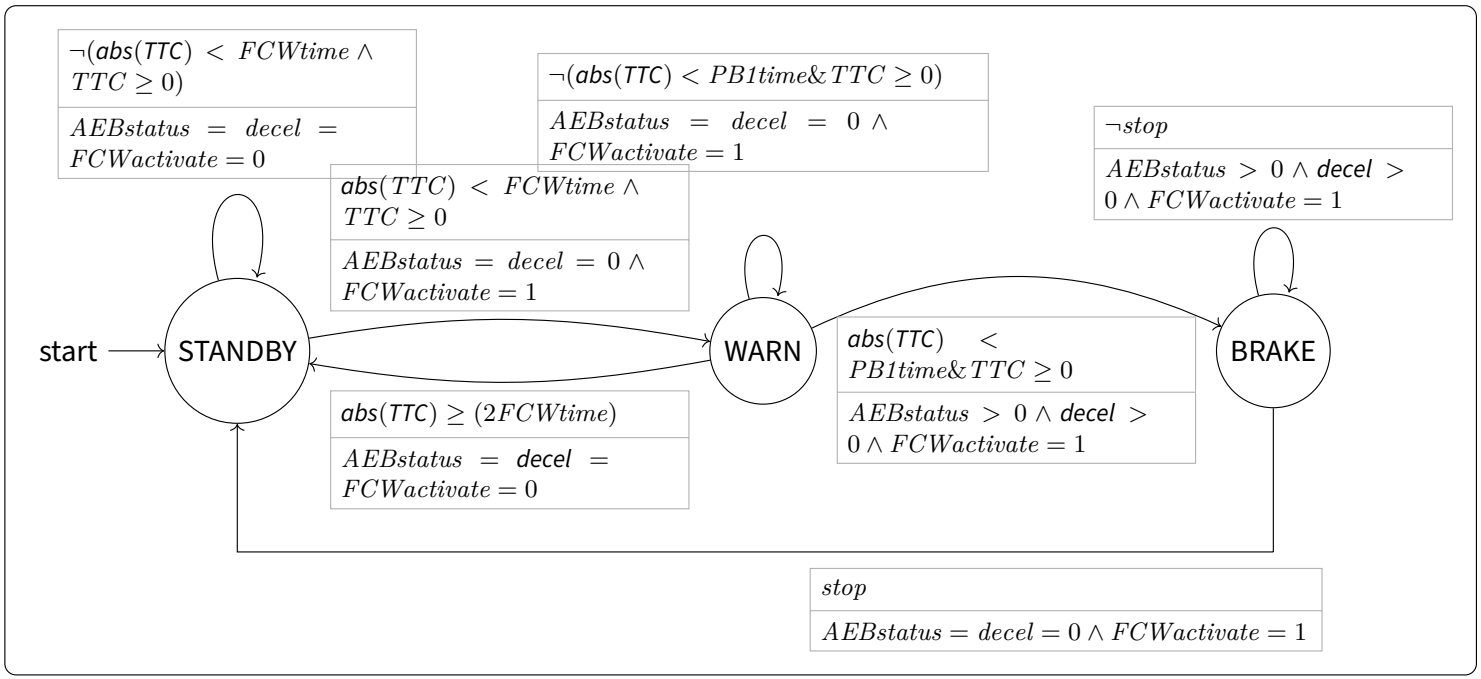


$\leq$



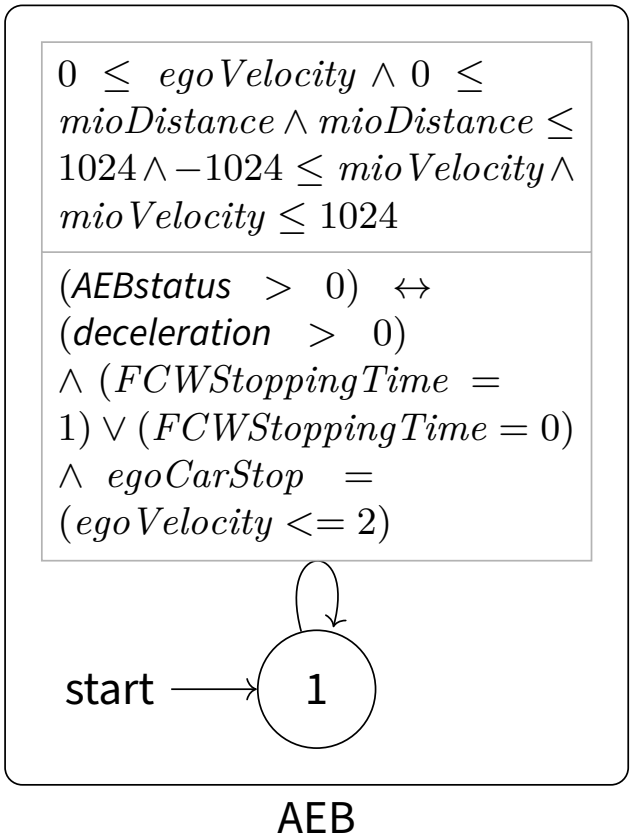
StoppingTimeCalculation

Subsystems II



AEB_Logic

Composed System



Results

- If car is not standing still and the system is engaged, the car is deceleratiing
- Interesting behavior mostly in AEB_Logic

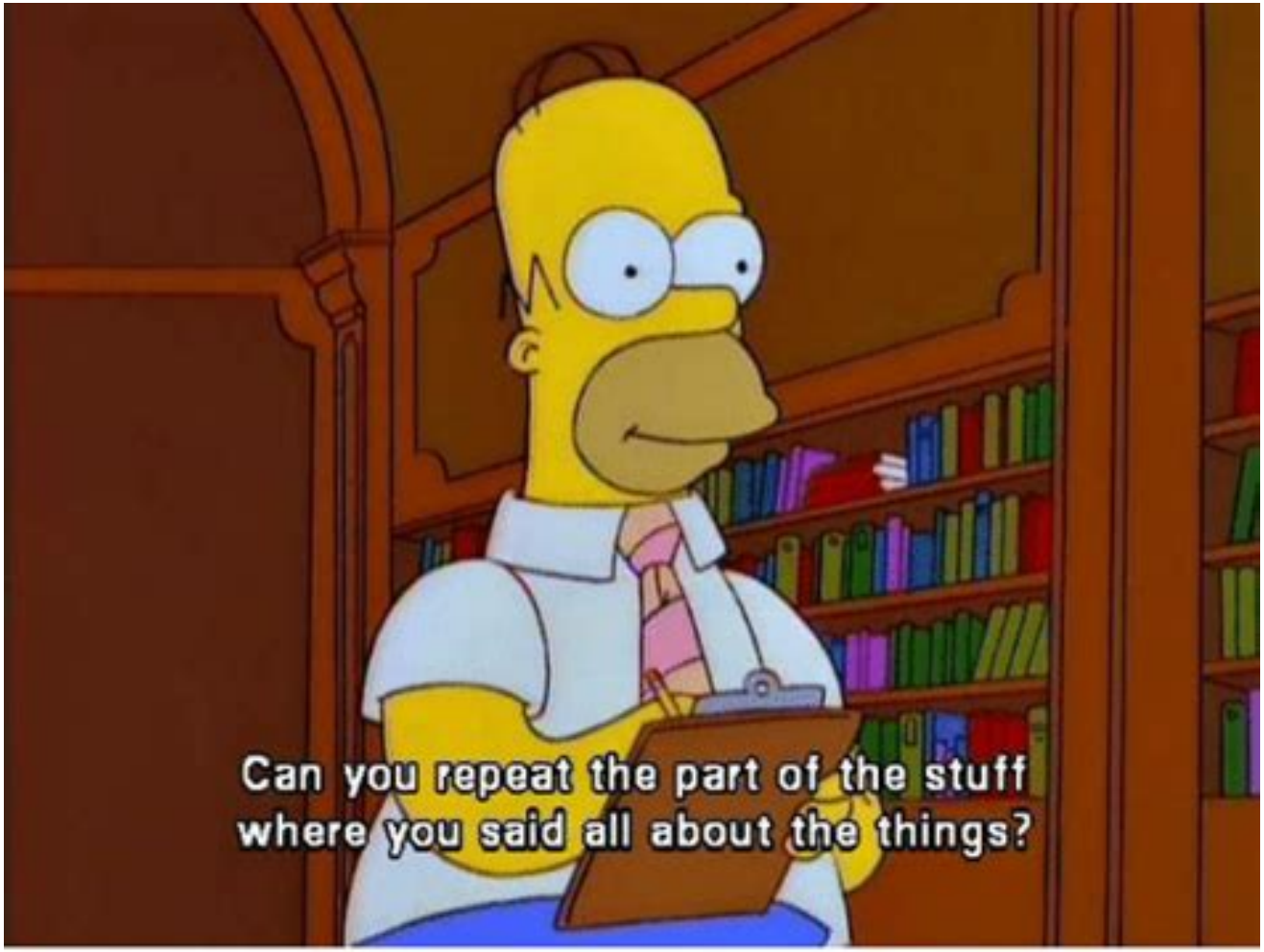
Conclusion

Contract Automata

- Mode-based reactive systems w/ Design-by-Contract
- Modular: Refinement & Composition
- Implementation and Case study available
- Submitted @ iFM’23

Future Work

- Specification Mining
- Runtime Verification
- Tooling (visualization, editing, etc.)


Can you repeat the part of the stuff
where you said all about the things?