Dimensions of language composition

June 14, 2022

Joeri Exelmans
Moussa Amrani

Why?

- Increasing complexity in CPSs

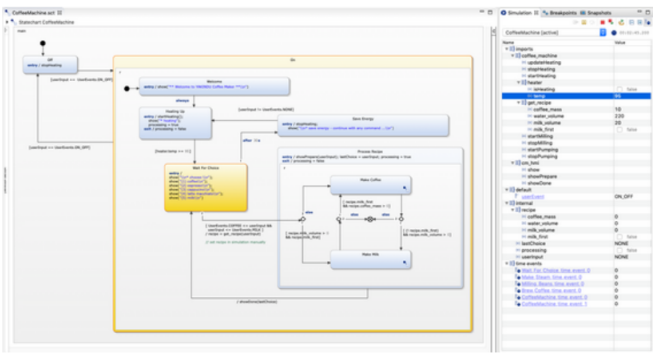
  => Growing need for hybrid languages

- Existing tools (e.g. SimuLink, YAKINDU) are developed in **ad-hoc** manner

  => immense effort to

  - create new hybrid variations
  - extend existing languages

  => Need **scientific foundations**
  + **tools** for hybrid language engineering



Scope of this presentation

- Description of dimensions of language composition variability
  - Not yet a feature model!
- Classification of existing approaches along these dimensions

Dimension: Composition of what?

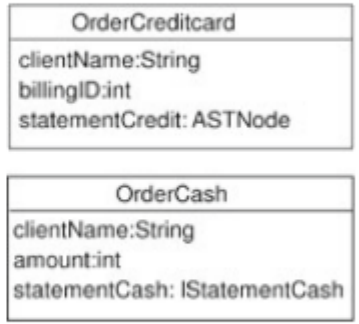
- **Concrete syntax**, plus **tooling**:
  - editing environment
  - simulation/debugging environment, ...
  - e.g. MPS, MontiCore (only textual)
- **Abstract syntax**
  - e.g. MPS, MontiCore, Melange, Sadaf’s paper
- **Semantics**
  - e.g. MPS, MontiCore, Melange, Sadaf’s paper

Example: CS + AS composition (MontiCore)

- grammar-centric
- generates:
  - CS (textual parser)
  - AS (set of Java classes)
- supports CS + AS embedding

MontiCore-Grammar

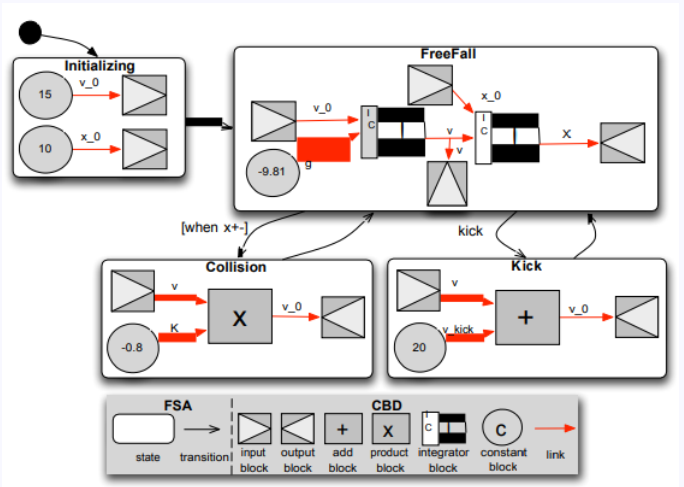
```
1  external StatementCredit;
2  external StatementCash
3    /example.IStatementCash;
4
5  OrderCreditcard implements Order =
6    "creditorder"
7    ID:IDENT Amount:NUMBER
8    StatementCredit;
9
10 OrderCash implements Order =
11   "cashorder"
12   ID:IDENT Amount:NUMBER
13   StatementCash;
```

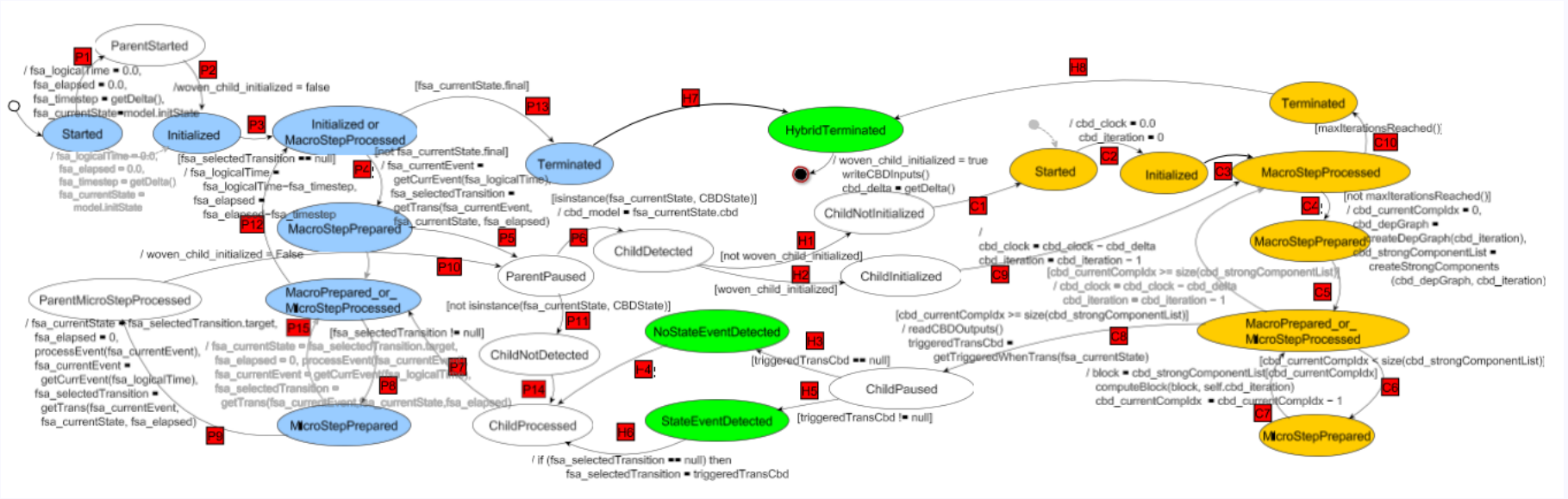


Dimension: Embedding vs. orthogonal composition

- **Embedding**
  - **Hierarchical** in nature: “master-slave”
  - CS: “Nesting” of CS elements (e.g. textual, visual, or “mixed”) and their editors (next slide...)
  - AS: Embedding pattern (next slide...)
  - Semantics: “Upper” language temporarily passes control flow to “lower” language
  - e.g. Mustafiz’ paper (AS + Sem.), Statecharts (CS + AS + Sem.)

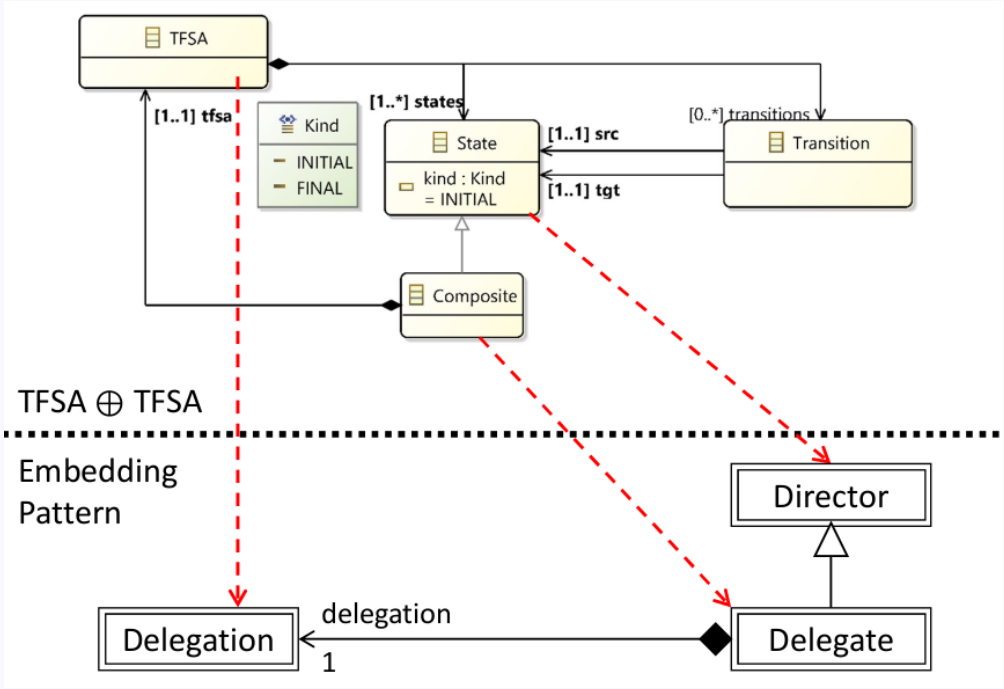
Example: Embedding of (operational) semantics



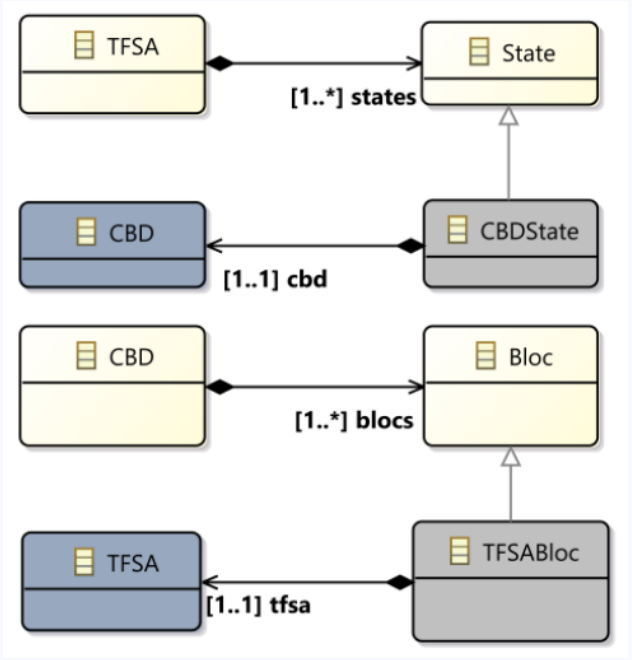


S. Mustafiz, C. Gomes, H. Vangheluwe and B. Barroca, "**Modular design of hybrid languages by explicit modeling of semantic adaptation,**" 2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS), 2016, pp. 1-8

Example: Abstract syntax embedding

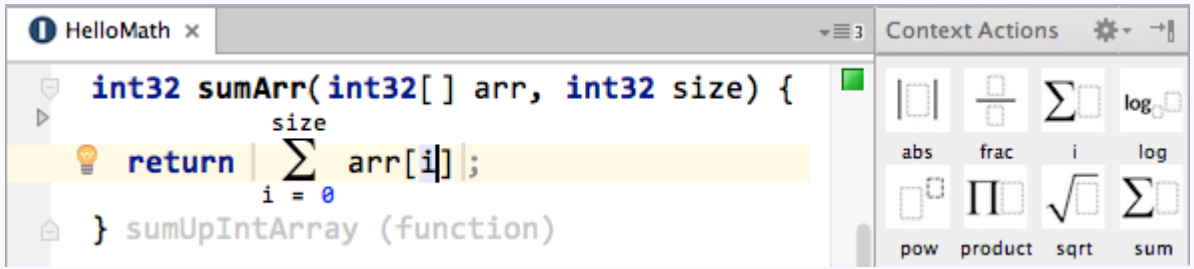


Embedding pattern applied to TFSA-within-TFSA



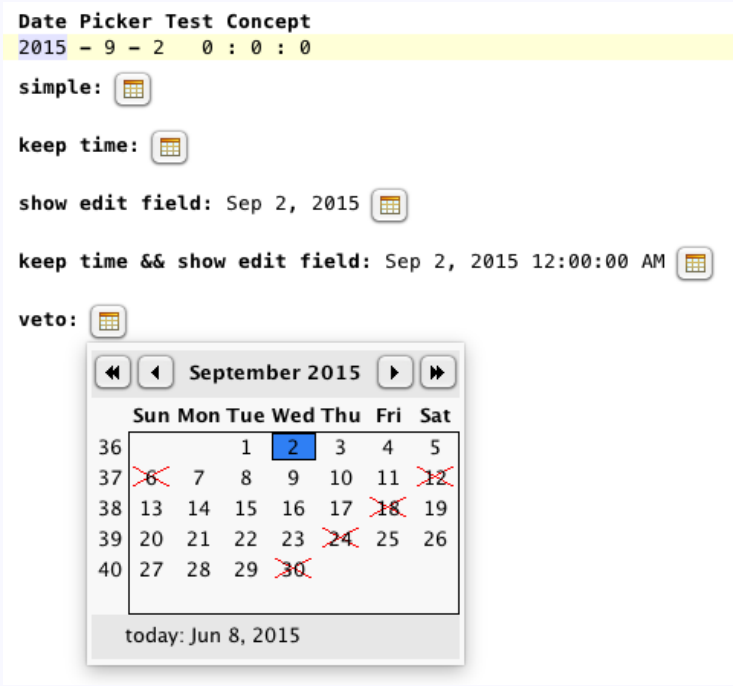
Embedding pattern applied to CBD-within-TFSA (top) and TFSA-within-CBD (bottom)

Example: CS embedding (MPS)



- Nesting of CS elements by “occupying space”
- Mixed textual / visual / tabular

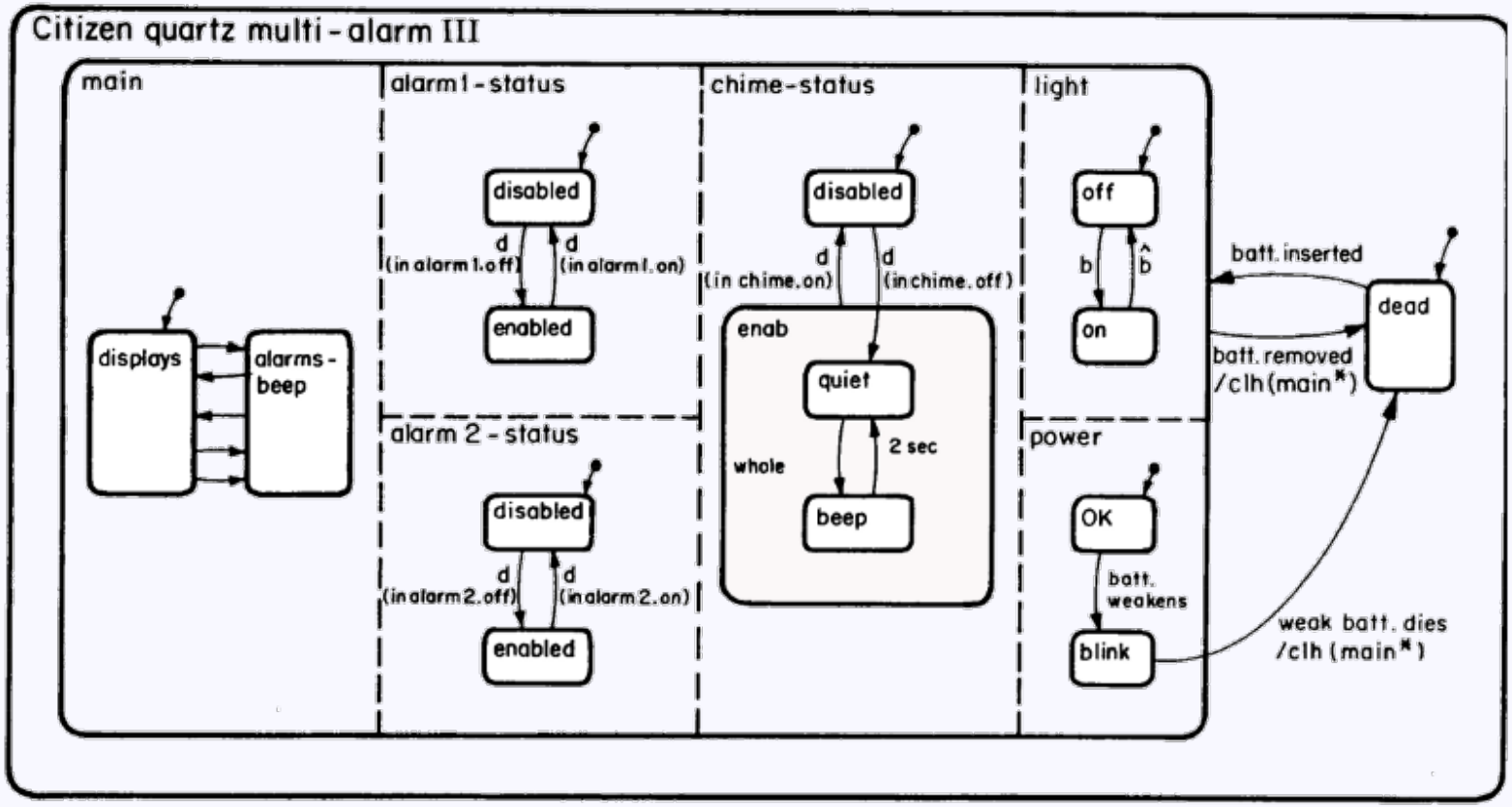
```
exported blockinterface IAdder [double arg1 / double arg2] => [double res]
  parameters [double dummy;]
  contract   [];

atomicblock SomeAdder realizes IAdder [double arg1 / double arg2] => [double res]
  parameters [double dummy;]
  ccode { res = 0; };
```



Dimension: Embedding vs. orthogonal composition

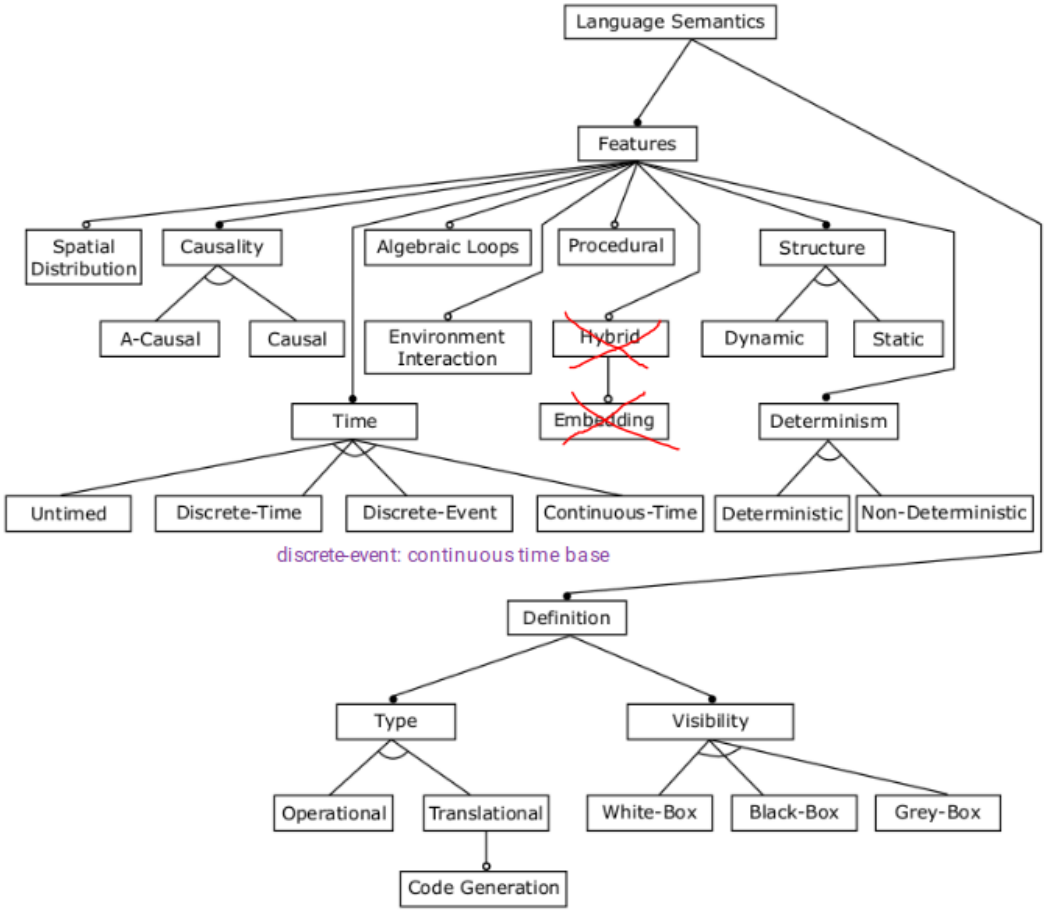
- **Orthogonal composition**
  - Composed languages are “equal”
  - (CS: merging of meta-models, merging of CS <-> AS mapping)
  - (AS: merging of meta-models)
  - Semantics: notion of **interleaving** by orchestrator
  - e.g. Co-simulation, Statecharts (orthogonal states)

Example: Embedding and orthogonal composition (Statecharts)

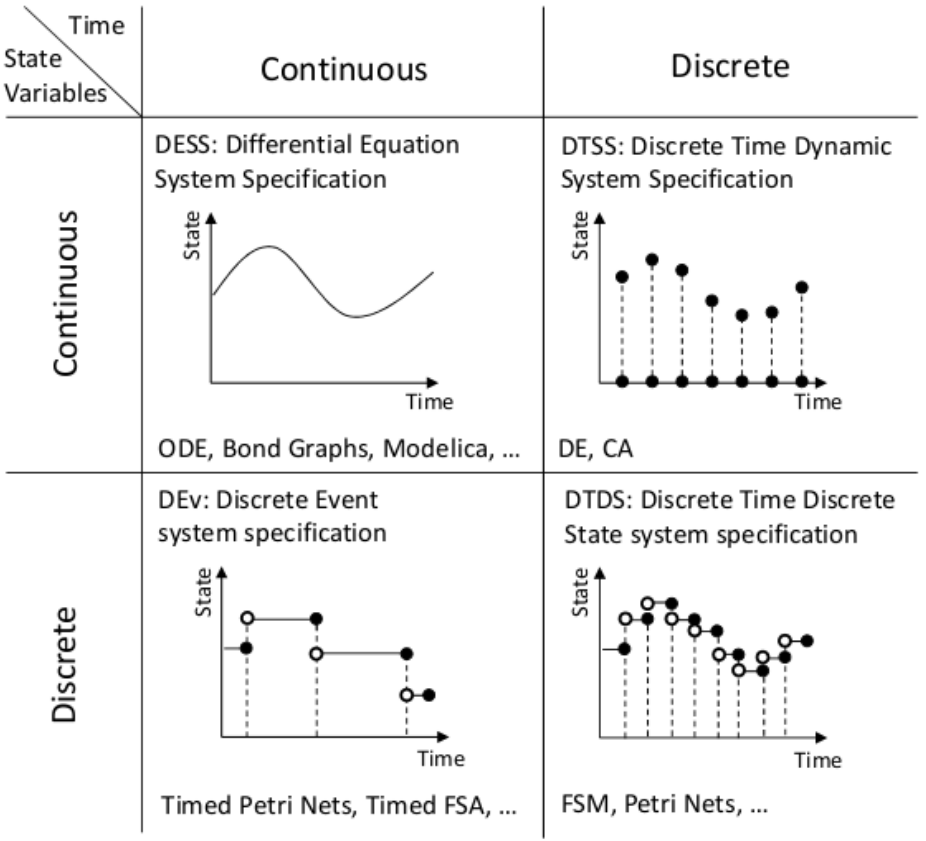


Dimension: Properties of composed languages

- **Same-formalism**
  - e.g. combining FSAs into Statecharts
- **Multi-formalism**
  - CS: composing **visual and textual CS**
    - e.g. Statecharts
  - Semantics: **complexity** of dealing w/ **multiple time bases**, ...
    - e.g. Sadaf's, Randy's paper



Dimension: Properties of composed languages



Zeigler, Bernard & Muzy, Alexandre & Kofman, Ernesto. (2018). Zeigler, B. P., Muzy, A., & Kofman, E. (2018). **Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations**. Academic Press

Dimension: Multi-abstraction (and multi-level)

- Multi-abstraction
  - Abstraction/refinement = substitutability wrt. properties of interest
  - Purpose: (1) performance (2) understandability/explainability
  - Statically: Same as AS embedding with **semantic continuity** (i.e. refinement should not alter existing behavior) ?

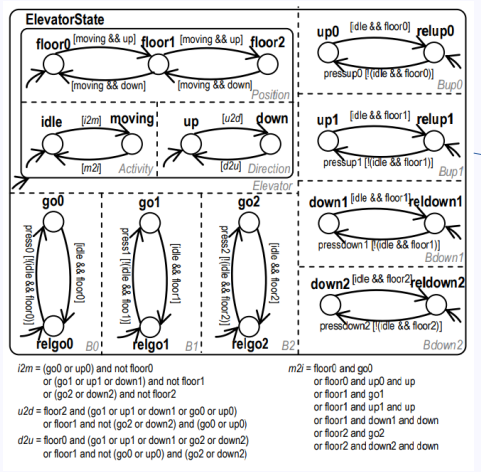
    e.g. Statecharts: composite states
  - Dynamically: At-runtime switching between refined/abstracted modes

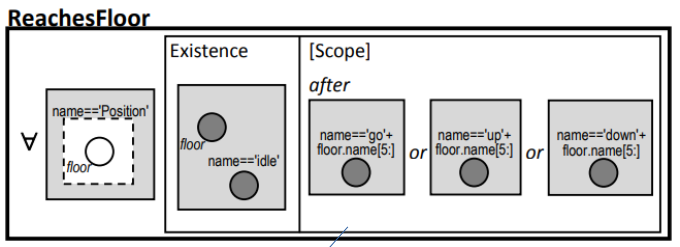
    e.g. Traffic simulation: Replacing individual cars by “traffic jam” object, and back
- (Multi-level)
  - Generalization of multi-abstraction to more generic ways of specifying “levels”

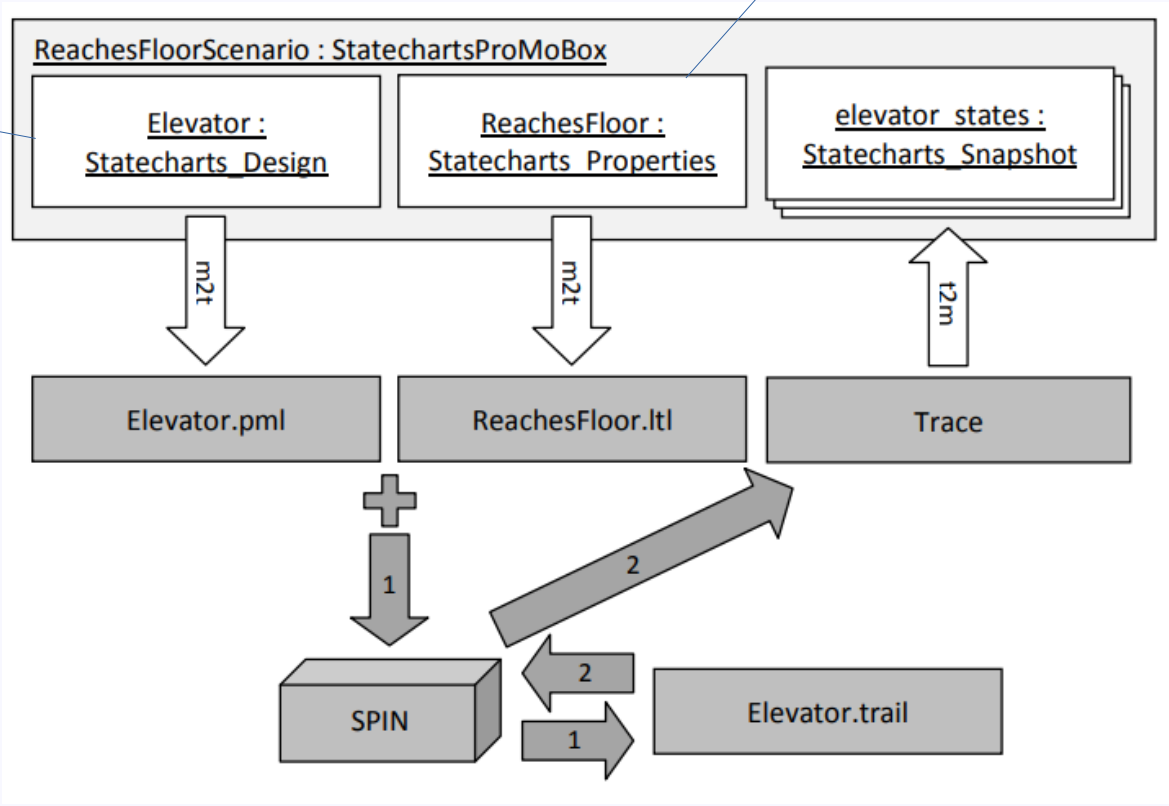
Dimension: Families of related languages

- For any DSL (w/ trace semantics), could explicitly model:
  - Design Language (what are valid models?)
  - Input Language (what are valid input traces?)
  - Runtime Language (what are valid runtime states/snapshots? what is the **initial** state of a model?)
  - Trace Language (what are valid (output) traces?)
  - Property Language (what properties can be validated on our models?)
- e.g. ProMoBox
  - Generates above 5 languages from a single DSL specification
- How to compose each these 5 languages?

Example: ProMoBox







Meyers, Bart & Wimmer, Manuel & Vangheluwe, Hans & Denil, Joachim. (2013). **Towards domain-specific property languages: The ProMoBox approach**. DSM 2013 - Proceedings of the 2013 ACM Workshop on Domain-Specific Modeling. 39-44. 10.1145/2541928.2541936.

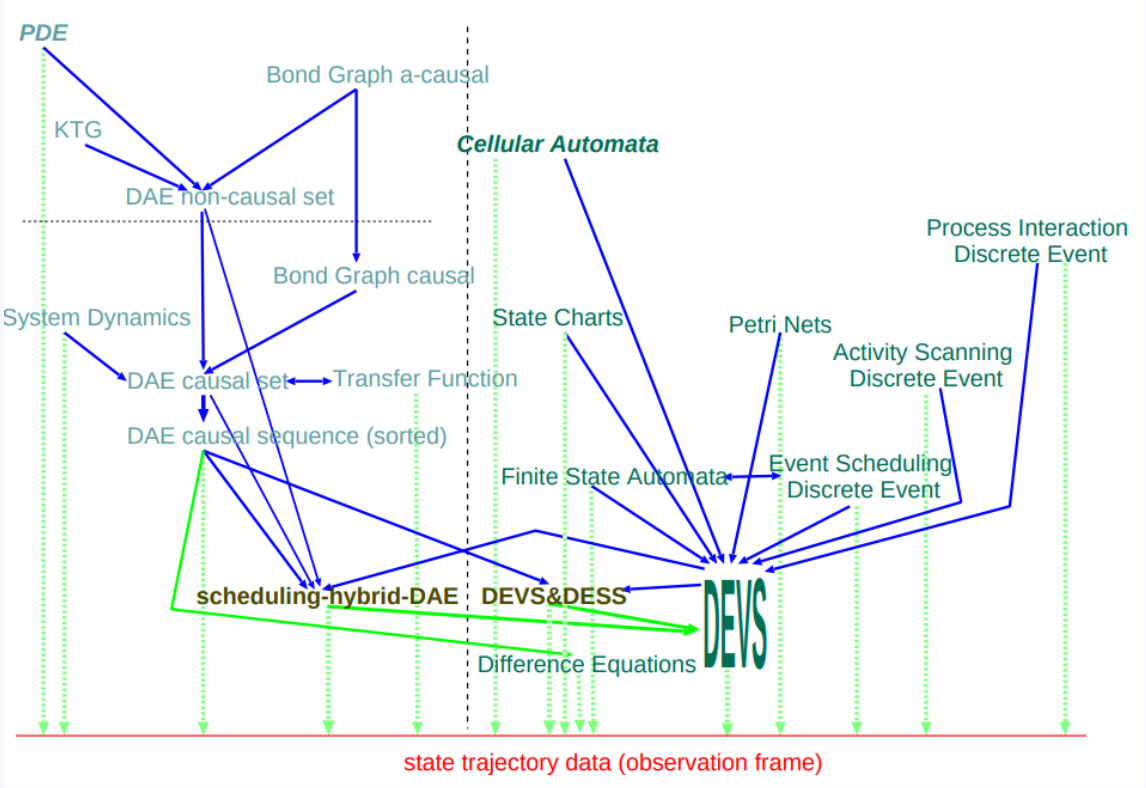
Dimension: Encapsulation

- **White-box**
  - Minimally constrained
  - AS: All model elements visible to and adaptable by “composer”
  - Sem: Interpreter internals (e.g. algorithm) visible to and adaptable by “composer”
  - e.g. Melange, Mustafiz’ paper (next slide…)
- **Black-box**
  - composition only through well-defined **interfaces**
  - AS: Only some (e.g. exported) model elements visible to “composer”
  - Sem: Only access via interface (e.g. “perform a step”)
  - e.g. Monticore, Co-simulation

Dimension: Operational vs. translational semantics

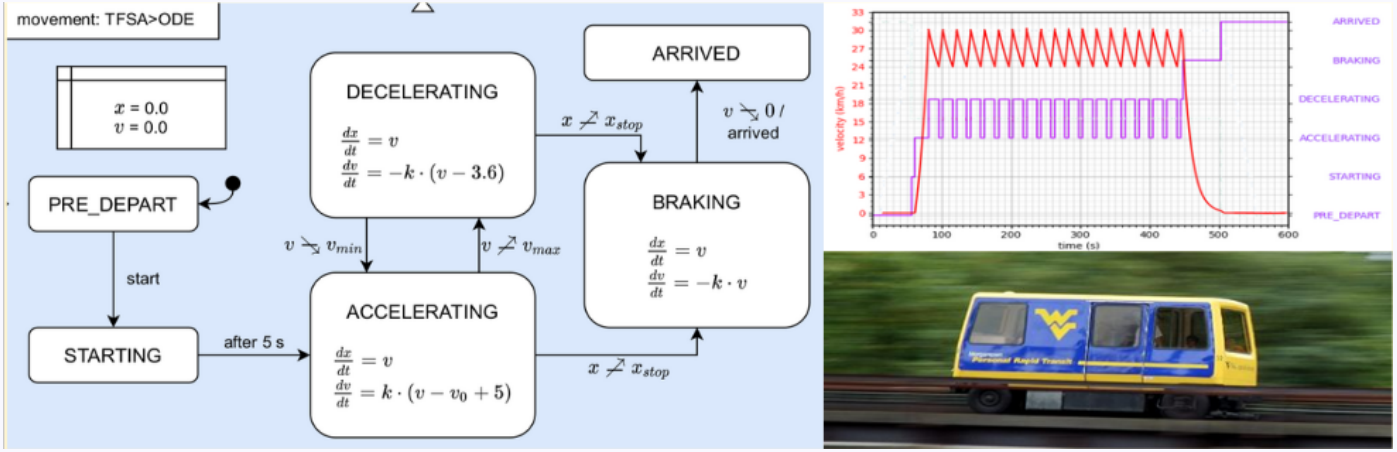
- **Operational**
  - Weaving of execution engines
  - Only for languages that have a **trace** semantics
  - e.g. Mustafiz' paper
- **Translational**
  - Mapping (different) languages onto a "common denominator" formalism (e.g. DEVS)
  - Always white-box (at level of AS)
  - Sub-dimensions:
    - can combine result of translation
      - e.g. Paredis' paper (next slide...)
    - can combine translations
      - e.g. ?
- *How to define "operational" and "translational"?*

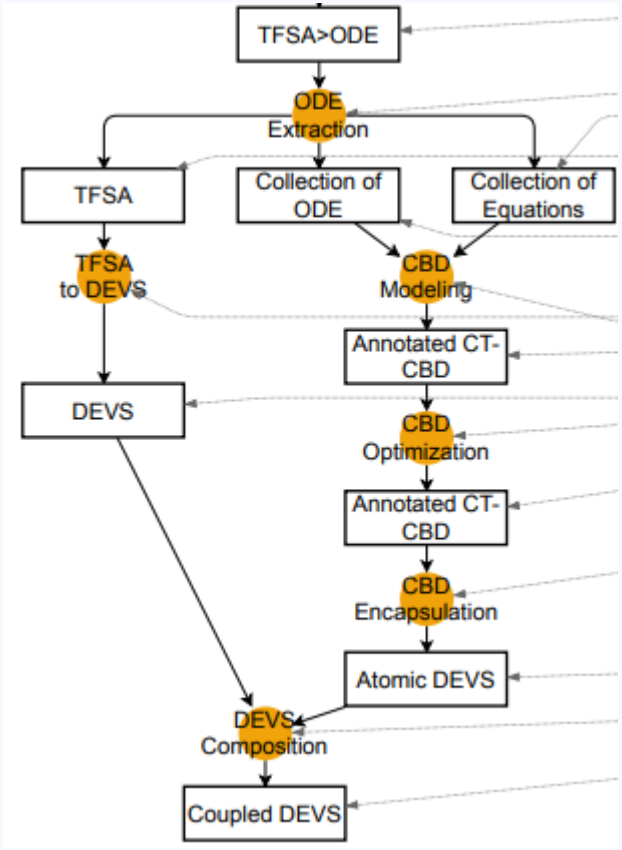
Example: Translational composition (DEVS as a “common denominator”)



Vangheluwe, Hans. (2000). **DEVS as a Common Denominator for Multi-formalism Hybrid Systems Modelling.** IEEE International Symposium on Computer-Aided Control System Design

Example: Translational composition (mapping ODE and TFSA onto DEVS)





R. Paredis, J. Denil and H. Vangheluwe, "**Specifying and Executing the Combination of Timed Finite State Automata and Causal-Block Diagrams by Mapping Onto Devs**," 2021 Winter Simulation Conference (WSC), 2021, pp. 1-12

Overview of dimensions

- “what”
  - CS (+ tooling) / AS / Sem.
  - Embedding / ortho comp.
  - Properties of composed languages (time base, ...)
  - Multi-level / multi-abstraction
  - Families of related languages
- “how”
  - Encapsulation (white-/black-box)
  - Operational / translational sem.

Legend

Composition property

Language property

Classification (non-exhaustive)

	type of example	**what:** CS / AS / semantics	**what:** embedding / orthogonal	**what:** time bases of composed languages	**how:** operational / translational	**how:** white-/ black-box
Mustafiz (CBD > TFSA)	demo of approach	AS + semantics	embedding	DE + CT	operational	white-box
Paredis (CBD > TFSA)	demo of approach	semantics	embedding	DE + CT	translational	white-box
MPS	language workbench	CS + AS + semantics	embedding	/	operational + translational	black-box
MontiCore	language workbench	(textual) CS + AS + semantics	embedding	/	operational	black-box
Co-simulation	technique	semantics	orthogonal	CT	operational	black-box
Statecharts (de- and re-constructed)	language	CS + AS + semantics	embedding + orthogonal	DE + timeless (action language)	?	?