Semantic aspects in Notation M
Environment Enabledness Memory
Determine transitions
Order of Small-Step
Aspect

1- Introduction
Different behavioural modeling languages are used in model-driven methodologies.
- How do we verify properties of models designed in different languages?
  - Solution: transform a design model notation to the input language of an analyzer, such as a model checker.
- Our contribution is a semantics-based translator from the family of big-step modeling languages (BSMLs) to the input language of SMV.
- A BSML is described using parameter values for semantic aspects.
- Using our translator, different combinations of options for semantic aspects will lead to generating new translators for specific languages.

2- Background: BSMLs
Big-Step: Reaction of model to an environmental input, which consists of a sequence of analysis steps.
- Many BSMLs exist with different syntaxes and semantics, such as Statecharts, Argos, Reactive Modules, and SCR, so we use a formalism syntax called CHTS.

3- Module Structure in SMV
One for each non-basic state, and includes:
- Enabledness flag for each transition in the state's scope
- An instance of the state modules corresponding to the state's direct children
- Two flags (macros) for the state enabledness and execution.

4- Translating the Structural Semantic Aspects
Concurrence?
MANY Find the inconsistent pairs of transitions based on Small-Step Consistency: R1
Find the inconsistent pairs of transitions in R1 based on Preemption: R2

SINGLE Insert invariants for each pair in R2 to disallow their concurrent execution
Invariants are always inserted in the state modules with tightest scope

5- Translating the Dynamic Semantic Aspects
Maximality
- Using a boolean variable to indicate when the big-step concludes and the next input should be read from the input module
- Event Lifeline
  - Relected in the next statement of events in the Snapshot module
- Assignment Memory Protocol and Enabledness Memory Protocol
  - If the option is Big-step, a copy of variable value at the beginning of the big-step is needed and will be used accordingly.
- Order of Small-Steps
  - If the option is Explicit invariants are used to impose that a transition is enabled, only if none of its predecessors are enabled.

6- Example
Traffic Light
North-South
- NS-Green, NS-Yellow, NS-Red
- EW-Green, EW-Yellow, EW-Red

East-West
- EW-Green, EW-Yellow, EW-Red

Module North-South(ss)
- basic states
- NS_Green, NS_Yellow, NS_Red
- EW_Green, EW_Yellow, EW_Red
- events
- end: change
- transitions
- t1_enabled : ss.NS_green & end;
- t2_enabled : ss.NS_yellow & end;
- t3_enabled : ss.NS_red & end;
- Def: INE
- North-South :=
- NS_Green | NS_Yellow | NS_Red;
- East-West :=
- EW_Green | EW_Yellow | EW_Red;
- priority invariant
- ss.t1_execute & ss.t2_execute & ss.t3_execute & ss.t4_execute;
- invariants
- ns1_enabled := ss.NS_green & ss.NS_yellow & ss.NS_red;
- ns2_enabled := ss.EW_green & ss.EW_yellow & ss.EW_red;
- SS: Main
- module snapshot(ss)
- ss1_enabled := ss1.Execute;