Step To Core Translation

Veritech Project
Declarations

- Type declarations
- Macro declarations
- Variable declarations
Variable Declarations

**Step**
- [exported] in, [exported] out, local
- Declaration in the modules only
- Non local variables are visible outside the module

**Core**
- Global variables are allowed
- All module variables are local
# Variable Declarations - Translation

<table>
<thead>
<tr>
<th>Step</th>
<th>external in</th>
<th>in</th>
<th>external out</th>
<th>out</th>
<th>local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>global variable + &quot;var' = var&quot; in assign field of ALL transitions inside the module where the &quot;in&quot; variable was defined.</td>
<td>global constant</td>
<td>global variable</td>
<td>global variable + &quot;var' = var&quot; in assign field of ALL transitions outside the module where the &quot;out&quot; variable was defined.</td>
<td>Usual module variables</td>
</tr>
</tbody>
</table>
Types

- Almost all translations are trivial
- Rational and Real type don’t exist in Core
- Tuple and records translated as variables

Example

Step  a: int*int*bool
Core   a_step_tuple_1:int
       a_step_tuple_2:int
       a_step_tuple_3:bool
Expressions

- Channel operation belong to SPL.
- Temporal operation not translatable
Fair Transition System

- Equivalent to Core program with single module.
- No fairness in Core
- Hold_previous are on
Module Transition System - Step

- Basic Module (interface and behavior)
- Abstract Module (interface only)
- Composed Module
  - Module composed from one or more instantiations of others modules
  - Step has many functions for module combination.
Module Transition System – Module Composition

1. Renaming module variables and transitions.
2. Hiding of module parameters and transitions.
3. Restriction – replacing parameters with new ones.
4. Augmentation - adding new output variables.
5. Casedistinction – behavior of composed module depend on input.

5. Module composition (||) between 2 or more module expressions is synchronic for exported transitions with the same name and asynchrony for the rest.
Module Transition System – Core

- Basic Modules
- Composed module
- No recursion
- No abstract modules
Module Transition System – Core – Module Composition

1. Possibility of transitions renaming in instantiation time
2. 3 types of module synchronization
   • Asynchrony – all transitions composed asynchronously
   • Synchronic – all transitions composed synchronically.
   • Partial synchronization – only listed transitions are synchronized
Module Transition System

Definitions:

1. Composition operations - renaming, hiding, augmentation, restriction.

2. Composition – combination of module expressions with “| |”.
Module translation

Translation of composed modules is divided to 2 parts

- Step To Step – after translation all composed modules consists of composition between instantiations of basic modules
- Step To Core
Module Translation – Step to Step

- Case Distinction – new module defined for each case, the expression is composition of these modules.
- Augmentation and Restriction - replaced by composition of original expression and instance of new module that has in the relation field the augmentation (restriction) expression.
Module Translation – Step to Step (Cont)

- Long compositions - module compositions that contain 3 or more module instances are replaced.
- Hiding and Renaming - its new versions of basic module contains the variable and of all expressions in the path between the EPR and the basic module are made.
Module Translation – Step to Core

- Every composed module has form $A1 \parallel A2$. ($A1$ and $A2$ are instantiations of basic modules)
- Pass over $A1..A2$ and make the Core partial synchronization, between modules with exported same named transitions and the Core asynchronous composition for the rest.
- Use the Core renaming for exported transitions to avoid name collisions
- Remove lists of exported transitions
Step To Core – General Translation Order

- Step to Step
  1. Change variables that need special translation (tuples, records) in all program space
  2. Module Step To Step translation
Step To Core – General Translation Order (Cont)

- **Step To Core**
  1. Module translation – Step to Core
  2. Variable declarations translation
  3. Types translation
  4. Transitions translation
Step To Core Translation – Examples

Examples of Step To Core translation are in StepToCore_2.ppt
Unresolved Problems

Recursion (exists in Step module composition, but not in Core)

There are two types of recursions:

a) Recursion that may be implemented as stack + loop

b) Recursion that may be implemented without stack by loop combination (For example: n!)