The Compositional Interchange Format: Introduction

Bert van Beek

Systems Engineering Group
Eindhoven University of Technology (TU/e)
http://se.wtb.tue.nl/sewiki/cif

25 November 2009
Purpose of the Compositional Interchange Format

Background

• For the development of high tech industrial systems many different formalisms and tools are required
• To ensure consistency of results across different models and tools, tool-integration and model reuse is essential

The main purpose of CIF is to

• Establish inter-operability of a wide range of tools by means of model transformations to and from CIF

In addition, tools that operate directly on the CIF are available for

• simulation of untimed, timed and hybrid CIF models
• supervisory control synthesis of untimed and timed models
CIF transformation examples

- **UPPAAL**
  - Timed automata verification

- **PHAVer**
  - Linear hybrid automata verification

- **Modelica, gPROMS**
  - CT plant design

- **Chi**
  - DE plant design

- **CIF**
  - ASCII plant specification
  - Graphical plant specification

- **CIF XML/Ecore controller**

- **CIF XML/Ecore controlled plant**
  - Timed/hybrid/realtime simulation

- **CIF XML/Ecore plant**

- **Rose RT Statecharts**
  - Real-time DE control

- **Matlab/Simulink**
  - CT controller design

- **Sequential Function Charts**
  - Graphical specification

- **Supervisory control tools**
  - DE controller synthesis

- **Modelica**
  - High speed hybrid simulation

CT = Continuous-time
DE = Discrete-event
Transformations without CIF

Language $A_0$  $\rightarrow$  Language $B_0$

Language $A_1$  $\rightarrow$  Language $B_1$

Language $A_2$  $\rightarrow$  Language $B_2$

Language $A_3$  $\rightarrow$  Language $B_3$
Transformations with CIF

- Language $A_0$
- Language $A_1$
- Language $A_2$
- Language $A_3$

- Language $B_0$
- Language $B_1$
- Language $B_2$
- Language $B_3$
Properties of CIF

- Based on automata
- *Formal and compositional semantics* allowing *property preserving model transformations*
- Differential algebraic equations (possibly discontinuous)
- *Hierarchy and modularity* to deal with *large scale systems*: model re-use, parallel models, and nested models
- Process interaction:
  - Communication via *shared channels*
  - Synchronization by means of *shared actions*
  - *Shared variables*
- Support for *urgency*:
  - urgent actions and channels
  - urgent locations
Development of CIF

Modelica, gPROMS
- DAE system simulation
- Optimization

Chi
- CT/DE simulation
- Process algebraic SCS

UPPAAL
- Timed automata verification
- Timed games SCS

PHAVer
- Linear hybrid automata verification

1. EU NoE HYCON1,2
2. EU FP7 Multiform
3. EU FP7 C4C
4. EU ITEA2 Twins
5. NL Darwin

Matlab/Simulink
- block oriented control

Sequential Function Charts control

Wonham event-based
- Modular SCS

Ma/Wonham state-based
- STS SCS

Real-time control
- Rose RT Statecharts: Error handling printer paper path
- Platform specific: MRI scanner patient support control

Switched linear systems interchange format

Discrete-time PWA
- Toolboxes: Multi-Param, Hybrid, Identification

SCS = Supervisory Control Synthesis
Industrial applications of CIF
Real-time error handling of paper path control using supervisory control synthesis
Industrial applications of CIF

Distributed control of the printing process using supervisory control synthesis

Océ specializes in durable, high-end equipment, suitable for corporate publishing/reproduction centers, as well as commercial printing and copying operations. Most equipment produced by Océ is high-speed (50 pages per minute and over) and has very high duty cycles (half a million pages a month and higher).

The process part, where the toner images are being formed and transferred to the paper, is a yet mostly unexplored area for control generation. Here we need to deal with heaters, temperature sensors, set-up and cleaning mechanisms, transport belts, etc.

Unlike the paper path, which is a coherent constellation of motors and pinches, this is a more diverse grouping of rather loosely coupled components.

The embedded control software for the process faces the problem that it has to control a very large amount of possibly conflicting machine states (like temperature too high while cleaning drum while solving error).

As illustration of undesired emergent behavior, consider the following example: It may happen that the imaging drum becomes too hot, in that case it should be cooled (by a fan), which can only happen in standby. However, a transition from running to
Industrial applications of CIF
Model based engineering of a document inserter
Industrial applications of CIF
Real-time control of a patient support system

Supervisory control synthesis using:
- modular supervisory control
- state-based supervisory control

Uncontrolled system: 6.3 billion states
Why use CIF for model transformations?

CIF is more general than other languages. E.g:

- UPPAAL has point to point synchronization by means of channels; PHAVer has multi-process synchronization by means of actions; CIF has both
- Block diagram languages (e.g. Matlab/Simulink) and hybrid IO automata deal with input-output systems; Behavioral languages (e.g. Modelica) deal with acausal systems; CIF deals with both
- Simulation languages deal with urgent (triggering guard semantics) and deterministic systems; Verification formalisms deal with non-urgent (using invariants to force actions) and non-deterministic systems; CIF deals with both
Why use CIF for model transformations?

CIF has a formal compositional semantics:

- The meaning of any CIF component is defined independently of its environment (bisimulation proven to be a congruence for all operators)
- E.g. if a hybrid CIF component $\alpha_{\text{hybrid}}$ with local variables can be simplified as an equivalent timed component $\alpha_{\text{timed}}$, then

$$\alpha_{\text{hybrid}} \parallel C$$

is equivalent to

$$\alpha_{\text{timed}} \parallel C$$

for all CIF components $C$
Different representations of CIF

Concrete CIF: User friendly syntax for modeling directly in CIF. Available in textual or graphical form

Abstract CIF: XML or ecore (for Eclipse platform) representation of CIF for model transformations

Core CIF: Smallest set of CIF primitives for mathematical definition of the formal semantics
Syntax of concrete CIF

Concrete CIF provides user-friendly syntax for the language elements from core CIF.

Furthermore, it extends core CIF with:

- *clocks* that are added for compatibility with timed automata,
- *input and output variables* that are added for compatibility with languages such as Simulink and PHAVer,
- *open and closed scopes* that allow the definition of variables, channels, clocks and actions as being local to facilitate hierarchy and modularity,
- *automaton definition and instantiation* that facilitate re-use of automata.
Bottle filling line example

\[ V_T, n, c, pH \]

\[ Q_{u}, c_u \quad Q_{a}, c_a \]

\[ Q_{F1} \quad Q_{Fr} \]
Bottle filling line: CIF graphical model

var alpha, beta: disc nat = 0, 0;
   n: cont real;
   pH: alg real = 7;
   c, Qa, Qu: alg real

physics
   inv
   n = c*V,
   dot V = Qu + Qa - QFl - QFr,
   dot n = cu*Qu + ca*Qa - c*QFl
       - c*QFr - Kloss*V,
   pH = - log c/1000,
   Qa = alpha*Qseta,
   Qu = beta*Qsetu

when pH >= 7.1
   do alpha:= 1
when pH <= 7
   do alpha:= 0

when V >= 10
   do beta:= 0
when V <= 2
   do beta:= 1

Bottle_Filling_Line bottles?
Bottle_Supply bottles!
Bottle_Filling_Line bottles?
Bottle filling line: CIF textual model

model Bottle_Filling_System =
| [ default urgent
  , connect {tank.V, left.VT, right.VT}
  , {tank.QFl, left.QF}
  , {tank.QFr, right.QF}
  , { bs.bottles
  , left.bottles
  , right.bottles
  }
:: tank:
  | [ output var V: cont real = 10
   extern var QFl, QFr: alg real
   intern var alpha: disc nat = 0
   ; beta: disc nat = 0
   ; n: cont real
   ; pH: alg real
   ; c, Qa, Qu: alg real
   init pH = 7
:: |( mode physics = eqn
   dot V = Qu + Qa - QFl - QFr
   , dot n = cu*Qu + ca*Qa - c*QFl
   - c*QFr - Kloss*V
   , n = c*V
   , pH = - log(c/1000)
   , Qa = alpha*Qseta
   , Qu = beta*Qsetu
:: physics
 )
  || left : Bottle_Filling_Line
  || right : Bottle_Filling_Line
  || bs : Bottle_Supply
:: closed
  || |( mode closed =
    when pH >= 7.1
      do alpha := 1 goto opened
    mode opened =
      when pH <= 7
        do alpha := 0 goto closed
    :: closed
  )
  || |( mode closed =
    when V <= 2
      do beta := 1 goto opened
    mode opened =
      when V >= 10
        do beta := 0 goto closed
    :: closed
  )
  || closed

Mathematical syntax of atomic interchange automaton

An atomic interchange automaton is a tuple

\[(V, v_0, flow, inv, tcp, E)\]

where

- \( V \): set of locations (vertices)
- \( v_0 \): initial location
- \( flow, inv, tcp \): functions from location to flow predicate, invariant, time-can-progress predicate, respectively
- \( E \): set of edges
- An edge can have a basic action label \( \ell \), a send action \( h!e \), or a receive action \( h?x \):
  - \((v, g, \ell, (W, r), v')\)
  - \((v, g, h!e, (W, r), v')\)
  - \((v, g, h?x, (W, r), v')\)
Mathematical syntax of core CIF

\[ \alpha ::= \alpha_{atom} \]

atomic interchange automaton

\| \alpha \parallel \alpha \]

parallel composition operator

\| \gamma_a(\alpha) \]

synchronising action operator \( a \) action label

\| u \gg \alpha \]

initialisation operator \( u \) predicate

\| [V x = e :: \alpha ] \]

variable scope operator \( x \) variable, \( e \) expression

\| [A a :: \alpha ] \]

action scope operator \( a \) action label

\| [H h :: \alpha ] \]

channel scope operator \( h \) channel

\| \partial_h(\alpha) \]

channel encapsulation operator \( h \) channel

\| U_z(\alpha) \]

urgency operator \( z \) action or channel

\| D_x:G(\alpha) \]

dynamic type operator \( G \) set of pairs of trajectories

\| st_x(\alpha) \]

state variable operator \( x \) variable

\| own_x(\alpha) \]

ownership operator \( x \) variable

Parallel composition now restrictive for synchronizing behavior. Other operators restrictive for all behavior.
Concluding remarks CIF tooling

- Translators available or under development for connecting the CIF to tools for:
  - Large scale DAE based hybrid system simulation
  - Hybrid system optimization
  - Verification of timed and hybrid systems
  - Supervisory control synthesis
  - Analysis and control of Discrete-time Piecewise Linear Affine systems
  - Block oriented analysis and (real-time) control system design

- Several tools available or under development for direct use of the CIF:
  - Graphical editor
  - Modeling and simulation
  - Hybrid system verification
  - Supervisory control synthesis and real-time control
Acknowledgements

- Dennis Hendriks, programmer SE
- Damian Nadalus, PhD student SE
- Jacobus (Koos) E. Rooda, group leader SE
- Ramon R.H. Schiffelers, postdoc SE