The Compositional Interchange Format
Concepts, Semantics and Applications

Bert van Beek

1 December 2010
Introduction

Background

- For the development of high tech systems many different modeling languages and tools are used.

Different models may exist of the same component, e.g. a logic controller:

- a *simulation* model of the controller and controlled system for performance analysis
- a *verification* model to check properties of the controller using a model checker
- an *implementation* model for real-time control in the actual system

How to avoid recoding of models in different languages, which leads to errors, loss of time and is expensive?

- model transformations
- co-simulation
Bi-lateral model transformations

- May lead to many model to model transformations \((m \times n)\)
- Developer of transformations must be familiar with many different formalisms
Model transformations via interchange format

- Only transformations to/from the interchange format \((m + n)\)
- Reduction of the implementation effort

Main requirements Compositional Interchange Format

- Generic and sufficiently rich modeling formalism
- Textual and graphical language and simulator for intuitive understanding of CIF models
- Conceptual language suited to model transformations
- Formal semantics
CIF language concepts

- Based on *automata*

- *Formal and compositional semantics* allowing
  - *property preserving model transformations*
  - *compositional verification*

- *Differential algebraic equations* (possibly discontinuous)

- *Large scale systems* modeling by orthogonal combination of:
  - parameterized process *definition* and *instantiation* (reuse, hierarchy)
  - *parallel composition*
  - and-or *superstates*

- *Process interaction*:
  - *Point to point communication*
  - *Multi-process synchronization*
  - *One-to-many broadcast communication*
  - *Shared variables*

- *Support for urgency*:
  - urgent actions and channels
  - urgent locations
Overview Compositional Interchange Format

Modelica, gPROMS
Muscod
• IPDAE system simulation
• Dynamic optimization

Chi
• CT/DE simulation

UPPAAL
• Timed automata verification

PHAVer, Ariadne
• Hybrid automata verification

CIF:
• Concepts
• Simulation
• External interfaces:
  Co-simulation
  Function calls
  EtherCAT fieldbus
• Refinement:
  And-Or superstates
  Stochastics
• CIF to CIF
  e.g.: CIF$_{hybrid}$ → CIF$_{timed}$
  CIF$_{AndOr}$ → CIF$_{flat}$
• Eclipse graphical editor

Switched linear systems interchange format

Cosimulation
• Matlab/Simulink and . . .

EtherCat real-time fieldbus

Logic controller design
• SFC, DC/FT

Supervisory Control Synthesis
• Event/State based

HW/SW codesign
• SystemC

Programmable Logic Controller
• IEC 61131-3 PLCopen

Industrial applications
• Twins: Printer paper path control: Rose RT Statecharts
• Darwin: MRI scanner patient support control
• C4C: Control of distributed printing processes

1. EU NoE HYCON2
2. EU FP7 Multiform
3. EU FP7 C4C
4. NL Darwin
5. EU ITEA2 Twins
6. EU NoE HYCON
Process interaction

Communication and synchronization

- Synchronization on shared action labels
- Sending and receiving information possible by means of actions $(a!e, a?x$, where $a$ is an action, $e$ an expression, $x$ a variable)

AND/OR synchronization

- AND synchronization: shared actions synchronize (e.g. multiple receivers receive the same message)
- OR synchronization: one of the shared actions synchronizes (as in channel communication in UPPAAL and Chi)

Normal or broadcast synchronization

- Normal: wait until synchronization/communication is possible
- Broadcast: the ‘receivers’ that are not ready do not participate in the synchronization/communication (broadcast as in UPPAAL)
Example: aerial vehicles and submarines (both unmanned)

Model of two aerial vehicles (only one shown) communicating with three submarines, by means of one shared action (frequency) $a$:

- OR synchronization between send actions of aerial vehicles
- Action $a$ is non-synchronizing for each aerial vehicle automaton, but synchronizing for the composition of the two automata
- Action $a$ is synchronizing for the three submarines. Action $a$ is declared as input (as in IO-automata)

This gives the UPPAAL broadcast channel semantics
Different flavors of CIF

For modeling

- Textual syntax aware editor
- Graphical editor

For definition of formal semantics

- Mathematical specification of core CIF
- Formal semantics defined using SOS (Structured Operational Semantics)

For model transformations

- Ecore class diagrams in Eclipse
CIF for modeling

```cif
model TankController()
  \{ cont control real V = 10.0
  ; var real Q1, Q0
  ; disc control nat n = 0
  \}
  \{ node physics = initial
    inv V' = Q1 - Q0
    Q1 = n * 5.0
    Q0 = sqrt(V)
  \}

Controller : \{ node closed = initial
  \(\text{when } V \leq 2 \text{ now do } n := 1 \) goto opened
  \(\text{when } V \geq 10 \text{ now do } n := 0 \) goto closed
  \}
```

Model TankController:
- cont control real V = 10.0
- var real Q1, Q0
- disc control nat n = 0
- Urgent false

Automaton Tank:
- physics
- initial
- inv V' = Q1 - Q0
- Q1 = n * 5.0
- Q0 = sqrt(V)

Automaton Controller:
- closed
- initial
- \(\text{when } V \leq 2 \text{ now do } n := 1 \)
- opened
- \(\text{when } V \geq 10 \text{ now do } n := 0 \)
Core CIF for formal semantics

An atomic interchange automaton is a tuple

$$(V, v_0, \text{flow}, \text{inv}, \text{tcp}, E)$$

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 $h?x$:
  - $(v, g, \ell, (W, r), v')$
  - $(v, g, h!e, (W, r), v')$
  - $(v, g, h?x, (W, r), v')$
Core CIF for formal semantics

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

- parallel composition operator \( \alpha \parallel \alpha \)
- synchronising action operator \( \gamma_a(\alpha) \)
- initialisation operator \( u \gg \alpha \)
- variable scope operator \([V \ x = e :: \alpha ]\)
- action scope operator \([A \ a :: \alpha ]\)
- urgency operator \(U_z(\alpha)\)
- dynamic type operator \(D_{x:G}(\alpha)\)
- control variable operator \(\text{ctrl}_x(\alpha)\)

• Orthogonal set of operators: separate operator for each concept
• Parallel composition *restrictive* for synchronizing behavior
• Other operators *restrictive* for all behavior
Compositionality

CIF has a formal compositional semantics (SOS):

- 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 $C$
And-Or superstate refinement in CIF

Any location of a automaton can be defined in terms of any other CIF components
CIF modeling extensions

CIF modeling extension extends core CIF with:

- *clocks* that are added for compatibility with timed automata,
- *input and output variables* that are added for compositional reasoning / compatibility with IO automata and with languages such as Simulink
- *automaton definition and instantiation* that facilitate re-use of automata

Semantics of modeling extensions of CIF formally defined by means of abstract grammars and functional transformations
Specification of transformations

- Bi-directional structure preserving transformations for the yellow subsets of the languages
- Model to model (M2M) transformation from source to target language of this subset
- Broaden the subsets by means of M2M transformations within each language
Transformation framework

CIF conceptual language definition based on

- Concepts and relations between concepts (class diagrams), e.g.
  - Concepts: automata, locations, edges
  - Relations: automaton contains locations and edges
- Invariants over these concepts (OCL constraints), e.g.
  - Within an automaton, the names of locations should be different
  - The source and target location of an edge belong to the same automaton as the edge itself

Transformation framework based on OMG standards

- Class diagrams for conceptual definition of CIF
- OCL: object constraint language
- QVT: model to model (concepts) transformation language
- M2T: model to text transformation language
CIF conceptual language definition: statements
CIF conceptual language definition: expressions
DSL transformation language

Modeling transformations using DSL transformation languages

- Transformations specified at problem domain instead of coding at implementation level
- Structure in transformations (re-use / chaining / hierarchy)
- Reducing implementation efforts
- Robust for changes

Implementation

- Tools currently implemented in the Eclipse Modeling Framework
Industrial applications of CIF
Paper path control and printing processes control using supervisory control synthesis
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

PICU:

Tabletop sensor (on/off)
Position encoder (on/off)
Horizontal brake (on/off)
Horizontal motor (on/off)
Clutch (on/off)
Max out sensor (on/off)
TTR button (on/off)
Vertical motor (up/down/stepped)
Vertical brake (on/off)
Max up sensor (on/off)
Max down sensor (on/off)

Light-visor
Tabletop
Bore
Patient support table
Conclusions on CIF itself

CIF generic and rich language

- Based on *automata*
- *Formal and compositional semantics*
- *Differential algebraic equations*
- *Large scale systems*: parameterized processes, and-or superstates
- *Process interaction*: point to point communication, multi-process synchronization, broadcast, shared variables
- Support for *urgency*

CIF tooling

- Textual and graphical editor
- Simulator
- Real-time control via EtherCAT
Conclusions on connecting CIF to other languages

Transformation framework based on OMG standards

- Class diagrams, OCL constraint language, QVT transformation language
- Specification of transformation at problem domain instead of coding at implementation level
- Implementation in Eclipse modeling framework

Connecting CIF via co-simulation

- Matlab/Simulink for control system design

Connecting CIF via model transformation to tools for among others

- Large scale DAE based hybrid system simulation and/or optimization
- Verification of timed and hybrid systems
- Supervisory control synthesis
- Real-time control on PLCs or via FPGA (future work)