The Compositional Interchange Format

concepts, developments, tools, applications

Bert van Beek

current results in cooperation with Pieter Collins (CWI), Dennis Hendriks, Albert Hofkamp, Damian Nadales, Koos Rooda, Ramon Schiffelers

Eindhoven University of Technology

http://se.wtb.tue.nl/sewiki/cif

2 April 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.
• Avoid the implementation of many bi-lateral translators between specific formalisms.

In addition, CIF

• Provides a generic modeling formalism and tools for a wide range of untimed, timed and hybrid systems.
Examples of translations

Without CIF:

Language $A_0$  
Language $B_0$

Language $A_1$  
Language $B_1$

Language $A_2$  
Language $B_2$

Language $A_3$  
Language $B_3$
Examples of translations

With CIF:

- Language $A_0$
- Language $A_1$
- Language $A_2$
- Language $A_3$
- CIF
- Language $B_0$
- Language $B_1$
- Language $B_2$
- Language $B_3$
CIF in EU projects

- Modelica, gPROMS
  - DAE system simulation
  - Optimization

- Hybrid Chi
  - Formal analysis
  - CT/DE simulation

- UPPAAL
  - Timed automata verification

- PHAVer
  - Linear hybrid automata verification

- EU NoE HYCON
  - EU FP7 Multiform
  - EU FP7 C4C
  - EU ITEA2 Twins

CIF:
- Modeling
- Simulation: symbolic, numeric, graphical output
- Comp Verification: Ariadne
- Distributed control
- Real-time control via EtherCat fieldbus
- CIF to CIF e.g: hybrid ⇒ timed

Switched linear systems interchange format

Discrete-time PWA
- Multi Parametric Toolbox
- Hybrid Toolbox
- Identification toolboxes

Matlab/Simulink
- block oriented control

Sequential Function Charts control

Wonham event-based
- Modular supervisory control synthesis

Ma/Wnh state-based
- STS Supervisory control synthesis

Rose RealTime
- Statecharts for real-time control
  - Error handling
  - printer paper path

Control via Firewire
- MRI scanner patient support control

Bert van Beek at DISC FP7
CIF: concepts, developments, tools, applications
Industrial application of CIF in the Twins project

Real-time error handling of paper path control using supervisory control synthesis
Industrial application of CIF in the C4C project

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 application of CIF in the Twins project

Model based engineering of a document inserter
Industrial application of CIF in the Darwin project

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
Requirements of CIF

- **Formal and compositional semantics** (bisimulation is a congruence for all operators) allowing **compositional (assume/guarantee) reasoning** and **property preserving model transformations**.
- Concepts based on **mathematics**, independent of implementation aspects such as equation sorting, and numerical equation solving algorithms.
- Support **arbitrary differential algebraic equations** (DAEs), including fully implicit equations, higher index systems, algebraic loops, steady state initialization, switched systems such as piecewise affine systems, and DAEs with discontinuous right hand sides.
- Support **hierarchy and modularity** to deal with **large scale systems**, by providing model re-use (parameterized model definition and model instantiation), parallel models, and nested models.
Requirements of CIF (cnt.)

• Support a wide range of *urgency* concepts, such as used in hybrid automata, including
  • urgency predicates
  • deadline predicates
  • triggering guard semantics
  • urgent actions.

• Support parallel composition with interaction by means of
  • CSP-like communication by means of *shared channels*
  • synchronization by means of *shared actions*
  • *shared variables*. 
Three formats of CIF

Abstract format: Facilitates mathematical definition of the formal semantics.

Concrete format: Provides user friendly syntax for the language elements of the abstract format
Suited for modeling directly in CIF.

Transfer format: Facilitates the file generation and parsing process.
Obsolete syntax of the abstract format of CIF

\[ \alpha ::= \alpha_{\text{atom}} \quad \text{atomic interchange automaton} \]

\[ \alpha \parallel \alpha \quad \text{parallel composition} \]

\[ \text{hide}_{\text{var}}(X_h, \alpha, \sigma_h) \quad \text{variable hiding operator} \]

\[ \text{hide}_{\text{act}}(L_h, \alpha) \quad \text{action hiding operator} \]

\[ \text{urgent}(L_u, \alpha) \quad \text{urgent action operator} \]

\[ \text{encap}(L_e, \alpha) \quad \text{action encapsulation operator}, \]

- \( L_h \): actions to hide,
- \( X_h \): variables to hide,
- \( L_u \): urgent actions,
- \( L_e \): actions to encapsulate.
Observate syntax atomic interchange automaton

\[ \alpha_{\text{atom}} ::= (X, X_i, \text{dtype}, V, v_0, \text{init}, \text{flow}, \text{inv}, \text{tcp}, L, E) \].

- \( X \): all variables, \( X_i \) internal variables.
- \( \text{dtype} \): function from variable to dynamic type: \textit{discrete}, \textit{continuous} or \textit{algebraic}.
- \( V \): locations, \( v_0 \): initial location, \( \text{init} \): initial condition.
- \( \text{flow}, \text{inv}, \text{tcp} \): functions from location to \textit{flow condition}, \textit{invariant}, \textit{time-can-progress condition}.
- \( L \): action labels, \( E \): edges.
Compositional formal semantics of CIF

The formal semantics are defined in terms of a *Hybrid Transition System*

Two types of transitions:

- discrete behavior by means of *action* transitions, labeled with action label and valuation.
- continuous behavior by means of *time* transitions, labeled with trajectories and duration.
Syntax of the concrete format of CIF

The concrete format consists of user-friendly syntax for the language elements from the abstract format. Furthermore, it extends the abstract format 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.
Example controlled tank in concrete CIF

Controlled tank system
Example controlled tank in concrete CIF

Simulation results

Simulation CIF model: tank.cif

Bert van Beek at DISC FP7  CIF: concepts, developments, tools, applications
Example controlled tank in concrete CIF

Iconic model using open scopes

model TankController
var V : cont real = 10
, alpha : disc nat = 0

controller
closed
when V <= 2
now
now
do alpha:= 1
do alpha:= 0
opened
tank
var Qin : alg real

physics
eqn Qin = alpha * 3 &
\dot V = Qin - 2
when V >= 10
now
now

Example controlled tank in concrete CIF

Textual model using open scopes

model TankController =
[[ extern var V : cont real = 10
  ; alpha: disc nat = 0
:: controller:
  |( mode closed = when V <= 2 now do alpha:= 1 goto opened
  , opened = when V >= 10 now do alpha:= 0 goto closed
  :: closed
  )|
|| tank:
  |( intern var Qin: alg real
  :: |( mode physics = eqn dot V = Qin - 2
  & Qin = alpha * 3
  :: physics
  )|
  )|]
Example controlled tank in concrete CIF

Iconic model using closed scopes

model TankController

controller
var valve : disc nat = 0

when VT <= 2
now
do valve := 1

when VT >= 10
now
now
do valve := 0

closed
opened

valve
VT

V
alpha

V

tank
var V : cont real = 10
, Qin : alg real

physics

eqn Qin = alpha * 3 &
dot V = Qin - 2
Example controlled tank in concrete CIF

Textual model using closed scopes

model TankController =
\[ connect \{tank.V, controller.VT\},\]
\{tank.alpha, controller.valve\}
:: controller:
\[ extern var VT : cont real \]
\{valve: disc nat = 0 \}
:: |( mode closed = when VT <= 2 now do valve:= 1 goto opened
\, opened = when VT >= 10 now do valve:= 0 goto closed
\:: closed
\) |
\]
\]| tank:
\[ extern var V : cont real = 10 \]
\{alpha: disc nat \}
\{intern var Qin: alg real \}
:: |( mode physics = eqn dot V = Qin - 2
\& Qin = alpha * 3
\:: physics
\) |
\] |
Bottle filling line
Bottle filling line

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

QFl: alg real [closed] [opened]
V: cont real
QFr: alg real

left
QF VT
Bottle_Filling_Line
bottles?

bs
Bottle_Supply
bottles!

right
VT QF
Bottle_Filling_Line
bottles?

done
model Bottle_Filling_System =
| [ connect {tank.V, left.VT, right.VT}
  , {tank.QF1, left.QF}
  , {tank.QF2, right.QF}
  , { bs.bottles
    , left.bottles
    , right.bottles
  }

:: tank:
  | [ output var V: cont real = 10
    extern var QF1, QF2: 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 - QF1 - QF2
    & dot n = cu*Qu + ca*Qa - c*QF1
      - c*QF2 - Kloss*V
    & n = c*V
    & pH = - log(c/1000)
    & Qa = alpha*Qseta
    & Qu = beta*Qsetu
    ]
:: physics
)||

|| |( mode closed =
    when pH >= 7.1
    now do alpha := 1 goto opened
    mode opened =
    when pH <= 7
    now do alpha := 0 goto closed
    :: closed
  )|
|| |( mode closed =
    when V <= 2
    now do beta := 1 goto opened
    mode opened =
    when V >= 10
    now do beta := 0 goto closed
    :: closed
  )|
||
 left : Bottle_Filling_Line
|| right : Bottle_Filling_Line
|| bs : Bottle_Supply
||
Bottle filling line
Bottle filling line

automaton Bottle_Filling_Line = |
|[ input var VT: real
    extern var QF: alg real
    chan bottles?: nat
    connect {QF, fp.QF},
    {VT, fc.VT},
    {bottles, fc.bottles}
    {fp.VB, fc.VB},
    {fp.gamma, fc.gamma},
    :: fc : Filling_Controller
|| fp : Filling_Physics
]]|

automaton Filling_Physics = |
|[ input var gamma: nat
    output var VB: cont real = 0
    extern var QF: alg real
    intern var QsetF: disc real = 1
    :: |( mode m = eqn dot VB = QF
        & QF = gamma*QsetF
        :: m
    )|
]]|
automaton Filling_Controller = |
  [ input var VB, VT: real
  output var gamma: disc nat
  extern chan bottles?: nat
  intern var n: disc nat = 0
  :: |( mode start =
      when n = 0 act bottles?n
      do (VB,gamma) := (0,1) goto filling
      when n > 0
      now do (VB,gamma) := (0,1) goto filling
  , filling =
      when VT <= 0.5
      now do gamma := 0 goto stopped
      when VB >= 1
      now do (gamma,n) := (0,n-1) goto start
  , stopped =
      when VT >= 0.7
      now do gamma := 1 goto filling
  :: start
  )| |
] |
automaton Bottle_Supply =
   |[ extern chan bottles!: nat
      intern clock t
   ::= |( mode m = when t >= 2 act bottles!24
         do t:= 0 goto m
   ::= m
   )|]
   |
Available CIF tools as delivered in HYCON

- CIF compiler that maps the concrete format to the abstract format, based on extended syntax of the concrete format of the CIF (including expressions).
- CIF stepper that generates the CIF transitions using:
  - Maple for symbolic solving and algebraic loop solving.
  - DDASRT for numeric solving.
- CIF simulator that provides a front-end to the stepper for single or exhaustive trace execution.
- CIF simulation visualization by means of Gnuplot.
- CIF abstract model visualization by means of Graphviz.
- Christian Sonntag (UNIDO):
  - SFC to CIF transformation and CIF simulation of resulting supermarket refrigeration system.
  - CIF to gPROMS transformation and simulation of resulting gPROMS model.
Overview of subset of current CIF tool chain

DE Simulation results → Simulator

Hybrid Simulation results → Simulator

Real-time simulation results → RT Simulator

Real-time control → RT Controller

$S_{R}.ads$ → TCT → $S.ads$ → ADS2CIF → $S.cif$ → CIF2PY → $S.py$

$P_{DE}.ads$ → ADS2CIF → $P_{DE}.cif$ → MERGECIF → $P_{DE}||S.cif$

$P_{HY}.cif$ → MERGECIF → $P_{HY}||S.cif$
Conclusions

• Basis of the *Compositional Interchange Format* has been developed in HYCON
• CIF has formal compositional semantics (bisimulation is a congruence for all operators)
• 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:
  • Modeling and simulation
  • Hybrid system verification
  • Supervisory control synthesis and real-time control