Static Analysis by Abstract Interpretation of communicating imperfectly-clocked Synchronous Programs

Julien Bertrane
bertrane@di.ens.fr

2 décembre 2006
System to analyse

SOFTWARE on synchronous hardware

HARDWARE (environment, sensors, actuators)
Difficulties and subsequent hypotheses

Framework includes realistic executions issues:

- Clock desynchronization allowed
- Non-constant delays during communications
- Graphical syntax

Simplifications:

- **Quasi-synchrony**: desynchronization: the cycle duration (period between two consecutive ticks) belongs to $[\alpha, \beta]$, $\alpha > 0$.

- Presently considered variables only booleans
- blackboard for synchronous units input
- Serial transmission between synchronous systems
- at initialization, all the “variables” are set to false
Goal: Automatic proofs of specifications

- **safety** specifications
  - For any behaviour \( s \), at any time \( t \), \( s(t) = \text{true} \)
Goal: Automatic proofs of specifications

- **safety** specifications
  - For any behaviour $s$, at any time $t$, $s(t) = true$

- **temporal** specifications
  For any behaviour $s$, there is no $t$ such that:
  
  $$\text{for any } t' \in [t, t+\alpha], s(t') = true$$
Goal: Automatic proofs of specifications

- **Safety** specifications
  - For any behaviour $s$, at any time $t$, $s(t) = true$

- **Temporal** specifications
  - For any behaviour $s$, there is no $t$ such that:
    
    \[ \forall t' \in [t, t + \alpha], s(t') = true \]

- **Quantitative** specifications
  - the outputs of 2 redundant systems match at least half the time of any interval of width $\delta$. 
Typical system: details of hardware hypotheses
Subsequent difficulties

- clock skew + delays in communications $\Rightarrow$ non denumerable set of behaviors

\[ \text{Synchronous system } C : \delta - \varepsilon, \delta + \varepsilon \]
\[ \text{Synchronous system } C' : \delta - \varepsilon, \delta + \varepsilon \]
Subsequent difficulties
Subsequent difficulties
Subsequent difficulties

- Proving specifications is difficult
- This is not the right way to handle redundancy
Behavior of a synchronous system

- A clock is a function $: \mathbb{N} \rightarrow \mathbb{R}^+$
- Clock parameter $: [\alpha, \beta]$, with $\alpha, \beta \in \mathbb{R}^+$ and $0 < \alpha \leq \beta$
- A clock $c$ satisfies $[\alpha, \beta]$ iff $c_{n+1} - c_n \in [\alpha, \beta]$
- $\text{DISCR}_{C_1}$ models the periodic reading of the input buffer
- $\text{SHIFT}_{C_1}$ models the waiting for the next clock tick, and the emission of its result at this next clock tick
Semantics: choices

- Continuous-time semantics instead of classical discrete one (PC, Message passing, ...)

- The semantics connects each point of control to a set of signals (i.e. element of $f : \mathbb{R}^+ \to \mathbb{B}$)

- A signal belongs to the semantics at point $p$ if there is a vector connecting each any point but $p$ to a signal compatible with $p$.

- If non-empty, the semantics often contains a non-countable infinity of signals.
Semantics of time-independent operators

\[ so_1(t) = \begin{cases} 
  \text{true} & \text{if } si_1(t) = \text{true} \\
  \text{or } si_2(t) = \text{true} & \\
  \text{false} & \text{else}
\end{cases} \]

\[ so_1 \triangleq \Psi_{\text{OR}}(si_1, si_2) \]
Semantics of time-dependent operators

\[ \alpha, \beta \] parameter of clock \( C \)

\[
so_1(t) = \begin{cases} 
\text{false} & \text{if } t < c(0) \\
\text{si}_1(c_n) & \text{if } t \in [c_n, c_{n+1})
\end{cases}
\]

\[ so_1 \triangleq \Psi_{\text{DISCR}_c}(si_1) \]
Syntax and semantics

**Syntax**

- `I_1` → `o_1`
- `o_2`
- `CONST α` → `I_1`
- `I_1` → `DELAY [α, β]` → `o_1`

**Semantics**

\[
\forall t \in \mathbb{R}^+, O_1(t) = l_1(t)
\]
\[
\forall t \in \mathbb{R}^+, O_2(t) = l_1(t)
\]
\[
\forall t \in \mathbb{R}^+, O_1(t) = \alpha
\]
\[
\forall t \in \mathbb{R}, O_1(t) = l_1(\delta(t))
\]
\[
\delta : \left\{ \begin{array}{l}
\exists \delta : \mathbb{R} \to \mathbb{R}, \text{monotonic}, \\
\forall t \in \mathbb{R}, \delta(t) - t \in [\alpha, \beta]
\end{array} \right.
\]
Syntaxe et sémantique
Syntaxe et sémantique

[0.;0.]

[0.4;0.5]

2nd Step

time: 0

C

C'

Ψ

DELAY

DISCR

DISCR

p 3 = p 0

p 4

p 5

p 6

p 7

D

Julien Bertrane
bertrane@di.ens.fr
Imperfectly-clocked Synchronous Programs

2 décembre 2006 15 / 43
Syntaxe et sémantique

[0.; 0.]

P_0

DISCR C

P_1

NOT

P_2

SHIFT C

P_3

C : 0.9, 1.1

[0.4; 0.5]

P_4

DISCR C

P_5

NOT

P_6

SHIFT C

P_7

D

3rd Step

time: 0

C_1

\[ \Psi_{\text{NOT}} \]

C : 0.9, 1.1

D

C' : 0.6, 0.7

\[ \Psi_{\text{NOT}} \]

time: 0

C_1'

\[ \Psi_{\text{NOT}} \]
Syntaxe et sémantique

[D]

4th Step

time: 0   C_1

P_3 = P_0

[t] [f] [f]

P_1

[t] [f] [f]

P_2

[t] [f] [t]

P_3

[t] [f] [f]

P_4

[t] [f] [f]

P_5

[t] [f] [f]

P_6

[t] [f] [t]

P_7

[t] [f] [f]

[0.4;0.5]

C : 0.9, 1.1

C' : 0.6, 0.7

[0.;0.]

DISCR C

NOT

SHIFT C

DISCR C

NOT

SHIFT C

P_0

P_1

P_2

P_3

P_4

P_5

P_6

P_7

D

C : 0.9, 1.1

C' : 0.6, 0.7

Ψ_{SHIFT}

Ψ_{DELAY}

Ψ_{SHIFT}

Julien Bertrane  bertrane@di.ens.fr  Imperfectly-clocked Synchronous Programs  2 décembre 2006  17 / 43
Syntaxe et sémantique

5th Step

time: 0

\[ \begin{array}{c}
\text{C}_0 \quad \text{C}_1 \quad \text{C}_2 \\
\hline
\text{P}_3 = \text{P}_0 & t & f & f & t \\
\text{P}_1 & f & f & f \\
\text{P}_2 & t & f & t \\
\end{array} \]

\[ \begin{array}{c}
\text{C}_0' \quad \text{C}_1' \quad \text{C}_2' \\
\hline
\text{P}_4 & t & f & f & f \\
\text{P}_5 & f & f & f \\
\text{P}_6 & t & f & t \\
\text{P}_7 & f & f & t \\
\end{array} \]
Syntaxe et sémantique

Modelisation  Semantics

[0.;0.]

P₀

DISCR C

P₁

NOT

P₂

SHIFT C

P₃

[0.4;0.5]

P₄

DISCR C

P₅

NOT

P₆

SHIFT C

P₇

D

C : 0.9, 1.1

C' : 0.6, 0.7

6th Step

time: 0

C₁

C₂

P₃ = P₀

f

f

t

P₁

f

f

P₂

f

f

t

time: 0

C₁' C₂' C₃'

P₄

f

f

f

P₅

f

f

f

P₆

f

f

t

P₇

f

f

t

[ D ]
- Not design: specification proof

- Difficulties: tricks
  - for robustness to desynchronization
  - for error recovery
  - for error robustness

- Idea of separation of design people and verification people

- Automatically generated code: classical patterns difficult to recognize

- Pattern may be simplified because classical academic tricks assume almost nothing

- Prototype and theory based on Abstract Interpretation

- Not complete: even safety undecidable
Abstract interpretation

- A set of elements
- $A^\#$ set of abstract elements
- $\alpha : A \rightarrow A^\#$
- $\gamma : A^\# \rightarrow A$
Abstract interpretation

- A set of elements
- $A\# \text{ set of abstract elements}$
- $\alpha : A \rightarrow A\#
- \gamma : A\# \rightarrow A$

- $A = \mathbb{Z}$
- $A\# = \mathbb{Z}/9\mathbb{Z}$
- $\alpha : x \mapsto x \mod 9$
- $\gamma : y \mapsto \{x, x = y \mod 9\}$
Abstract interpretation

- A set of elements
- \( A^\# \) set of abstract elements
- \( \alpha : A \rightarrow A^\# \)
- \( \gamma : A^\# \rightarrow A \)

- \( A = \mathbb{Z} \)
- \( A^\# = \mathbb{Z} / 9\mathbb{Z} \)
- \( \alpha : x \mapsto x \mod 9 \)
- \( \gamma : y \mapsto \{ x, x = y \mod 9 \} \)

- \( +^\#(4 \mod 9, 6 \mod 9) = 1 \mod 9 \)
- \( \text{if } \Psi \circ \gamma \subseteq \gamma \circ \Psi^\# \)
- \( \text{gfp } \Psi \subseteq \gamma(\text{gfp } \Psi^\#) \)
Abstract interpretation based analysis

- $[D]$ is the semantics of a set $D$ of systems.
- $[P]$ is the set of behaviors satisfying a property $P$.

**Former goal**: Prove that $[D] \subseteq [P]$.

**Now**: $(\Psi \cap Id)([D] \cap [\neg P]) \subseteq [D] \cap [\neg P]$

**Thus**:

$$[D] \cap [\neg P] \subseteq \text{gfp}_{[\neg P]}(\Psi \cap Id) \subseteq ? \emptyset$$

**True if (not iff)**:

$$\text{gfp}_{[\neg P]}(\Psi \cap Id) \subseteq ? \emptyset \dashv \dashv$$
1st abstract domain

A constraint $\exists [a; b] : x$ forces signals to be equal to $x$ at least once during $[a; b]$.

A constraint $\forall \langle a; b \rangle : x$ forces signals to be equal to $x$ during the whole $[a; b]$. 
Abstract Operators and Constraints: an example

\[ \begin{align*}
\psi_\#_{\text{DELAY}[\alpha,\beta]}(\exists [a; b] : x) & \triangleq \exists [a - \beta; b - \alpha] : x \\
\psi_\#_{\text{DELAY}[\alpha,\beta]}(\forall \langle a; b \rangle : x) & \triangleq \forall \langle a - \alpha; b - \beta \rangle : x
\end{align*} \]
Abstract Operators and Constraints: an example

\[ \text{DISCR} [\mu, \nu] \]

\[ \text{DISCR} [\mu, \nu] \]

\[ \psi^\#_{\text{DISCR}[\mu, \nu]} (\exists [a; b] : x) \rightleftharpoons \exists [a - \nu; b] : x \]

\[ \psi^\#_{\text{DISCR}[\mu, \nu]} (\forall \langle a; b \rangle : x) \rightleftharpoons \bigwedge_{t \in [a,b]} \exists [t - \nu; t] : x \]
Iterating up to a fixpoint
Iterating up to a fixpoint
Iterating up to a fixpoint
Iterating up to a fixpoint
Iterating up to a fixpoint

\[ \bigwedge [t-41, t] : True \]
\[ \delta-39 < t < \delta+59 \]

\[ \langle \delta-39, \delta+59 \rangle : \text{False} \]
Iterating up to a fixpoint

\[ \Delta [t-41,t]: False \]
\[ \delta-39 < t < \delta+59 \]
Iterating up to a fixpoint

\[ \land [t-51, t]: \text{False} \]
\[ \delta - 39 \leq t \leq \delta + 59 \]
Iterating up to a fixpoint

\[ \Delta [t-92, t-39]: \text{False} \]

\[ \delta-39 < t < \delta+59 \]
Iterating up to a fixpoint
Example: Result of the analysis

Hence no behaviour can satisfy at control point $t_4$:

$$
\langle \delta - 39, \delta + 59 \rangle : True \land \bigwedge_{\delta - 39 \leq t \leq \delta + 59} ([t - 92, t - 39] : False)
$$

because it implies $[\delta - 33, \delta + 20] : False$
Weaknesses of the Constraints domain

- Weak loss of precision in the case of: DELAY, DISCR, SHIFT, NOT,
- Unwished loss of precision in the case of: AND, OR, XOR

∀⟨0; 5⟩ : false
2nd Abstract Dom. : Changes Counting Dom.

\begin{itemize}
  \item width=$\delta$
  \item $\# \text{ value chng} \leq 5$
\end{itemize}

\begin{itemize}
  \item width=$\delta$
  \item $\# \text{ value chng} \leq 5$
\end{itemize}
Time-dependent Abstract Operators inside the Changes Counting Domain

\[ \alpha \]
\[ \beta \]

- \([\alpha, \beta]\) parameter of clock \(C\)
- \(\Psi_{\text{DISCR}[\alpha, \beta]}(-) \equiv (1, \alpha)\)
Time-independent Abstract Operators inside the Changes Counting Domain

\[ \overrightarrow{\psi}_\text{AND}(((n_1, \delta_1), (n_2, \delta_2)) \uparrow (\tilde{n}_1 + \tilde{n}_2, \tilde{\delta}_1) \]
Analysis Abstract interpretation

Time-independent Abstract Operators inside the Changes Counting Domain

\[ \Psi \rightarrow (n_1, \delta_1) \land (n_2, \delta_2) \triangleq (\tilde{n}_1 + \tilde{n}_2, \tilde{\delta}_1) \]

\( \varphi \) is a reframing function and
Time-independent Abstract Operators inside the Changes Counting Domain

\[ \Psi_{\text{AND}}((n_1, \delta_1), (n_2, \delta_2)) \triangleq (\tilde{n}_1 + \tilde{n}_2, \tilde{\delta}_1) \]

- \( \varphi \) is a reframing function and
- \( \varphi((n_1, \delta_1), (n_2, \delta_2)) = ((\tilde{n}_1, \tilde{\delta}_1), (\tilde{n}_2, \tilde{\delta}_1)) \).
Reduced Product Constraints-Changes Counting Domain

\[ \text{width} = \delta \]
\[ \# \text{ value chng} \leq 1 \]
Reduced Product Constraints-Changes Counting Domain

- \[ \begin{array}{c}
\text{width}=\delta \\
\text{value chng} \leq 1
\end{array} \]

Julien Bertrane  bertrane@di.ens.fr  Imperfectly-clocked Synchronous Programs  2 décembre 2006  41 / 43
3rd abstract domain: Integral bounding Dom.

- Express quantitative properties
Conclusion

- Realistic (?) model of execution of imperfectly-clocked communicating synchronous systems (imperfect clocks, non-instantaneous delays, blackboard)

- Syntax includes annotations about hardware imperfections

- Semantics mixes discrete/continuous notions, but mainly continuous-time

- Analysis retrieves discrete behaviors (Value changes counting)

- Need for a better knowledge of robustness/redundancy techniques