Components, Interfaces and Compositions: from SoCs to SOCs

Partha S. Roop

University of Auckland

Organization

- Significance of components and interfaces.
- Two recent frontiers – SoCs and SOCs.
- Key problems:
  - Component matching – refinement based and DES control based.
  - Component composition – converter / choreographer synthesis.
- Conclusions.

Acknowledgements

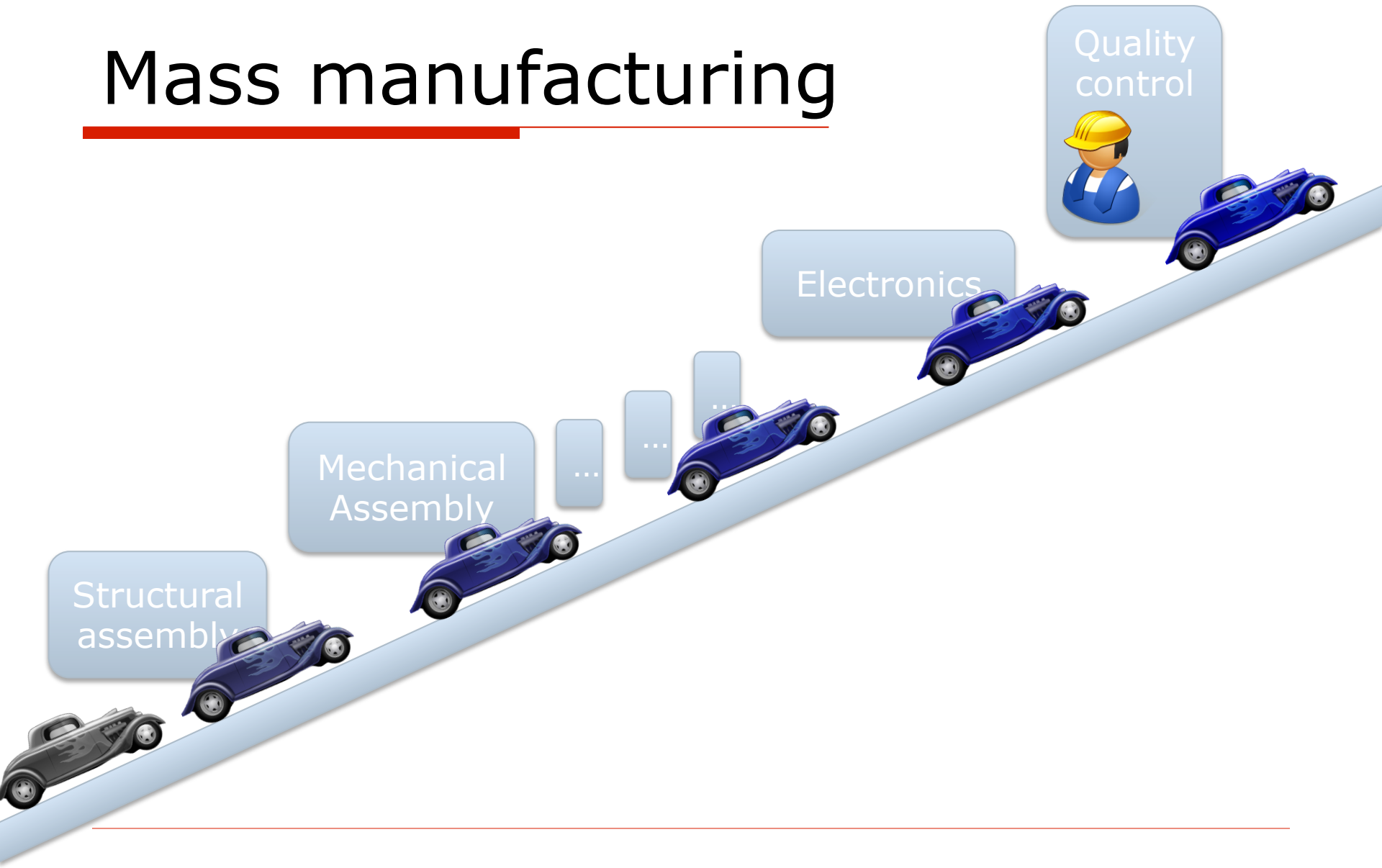
- Forced Simulation is joint work with A. Sowmya (UNSW), S. Ramesh (General Motors R&D) and the link to DES is with Robi Malik (Waikato).
- Local module checking and converter synthesis is joint work with Roopak Sinha (Postdoc) and Samik Basu (Iowa State).
- Web Services composition is joint work with Adeel Ali (PhD student), Ian Warren (Soft. Eng., Auckland) and Zeeshan Bhatti (PhD student)

I, Pencil

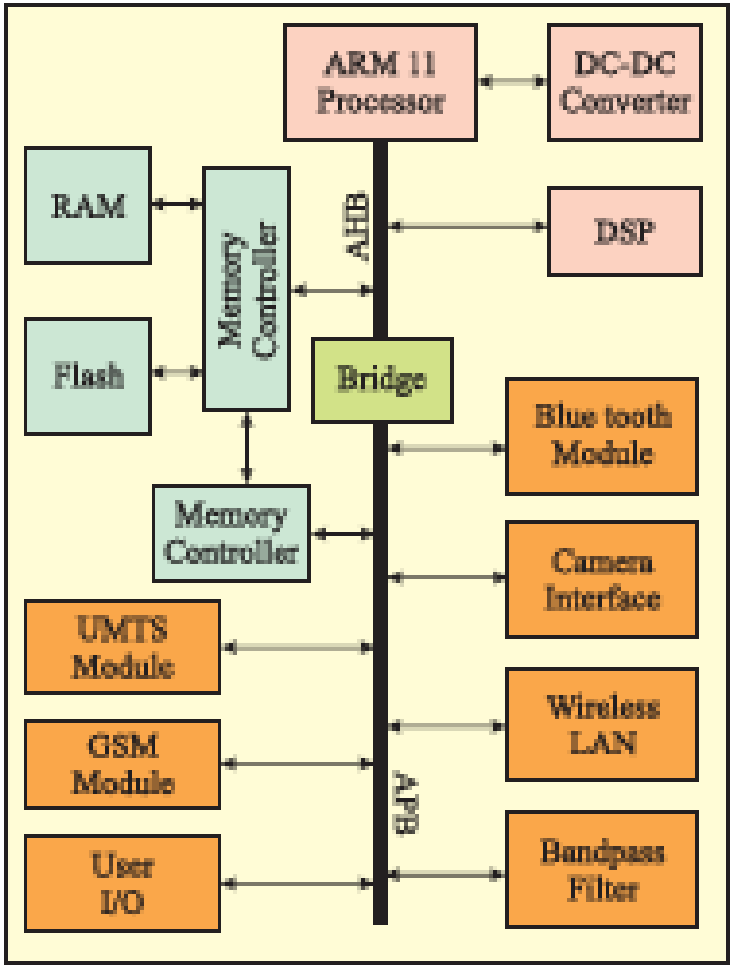
- "Simple? Yet, *not a single person on the face of this earth knows how to make me*."
  - Making of lead *(graphite + clay)*
  - Making of body *(cedar + lacquer)*
  - Eraser *(rubber + factice + …)*
  - Label *(carbon + resin + …)*
  - Ferrule *(brass + zinc + …)*

"I, Pencil", Leonard E. Read (1898-1983), published Dec 1958, issue of *The Freeman.*

Mass manufacturing



A System-on-a-chip (SoC) Example



Source: R. Sinha, Automated Techniques for SoC Verification, PhD thesis, University of Auckland, 2008.

Consumer electronics revolution fuelled by SoCs

Embedded Systems





Safety-critical concerns



- Compliance to strict safety standards [IEC 61508, DO 178]

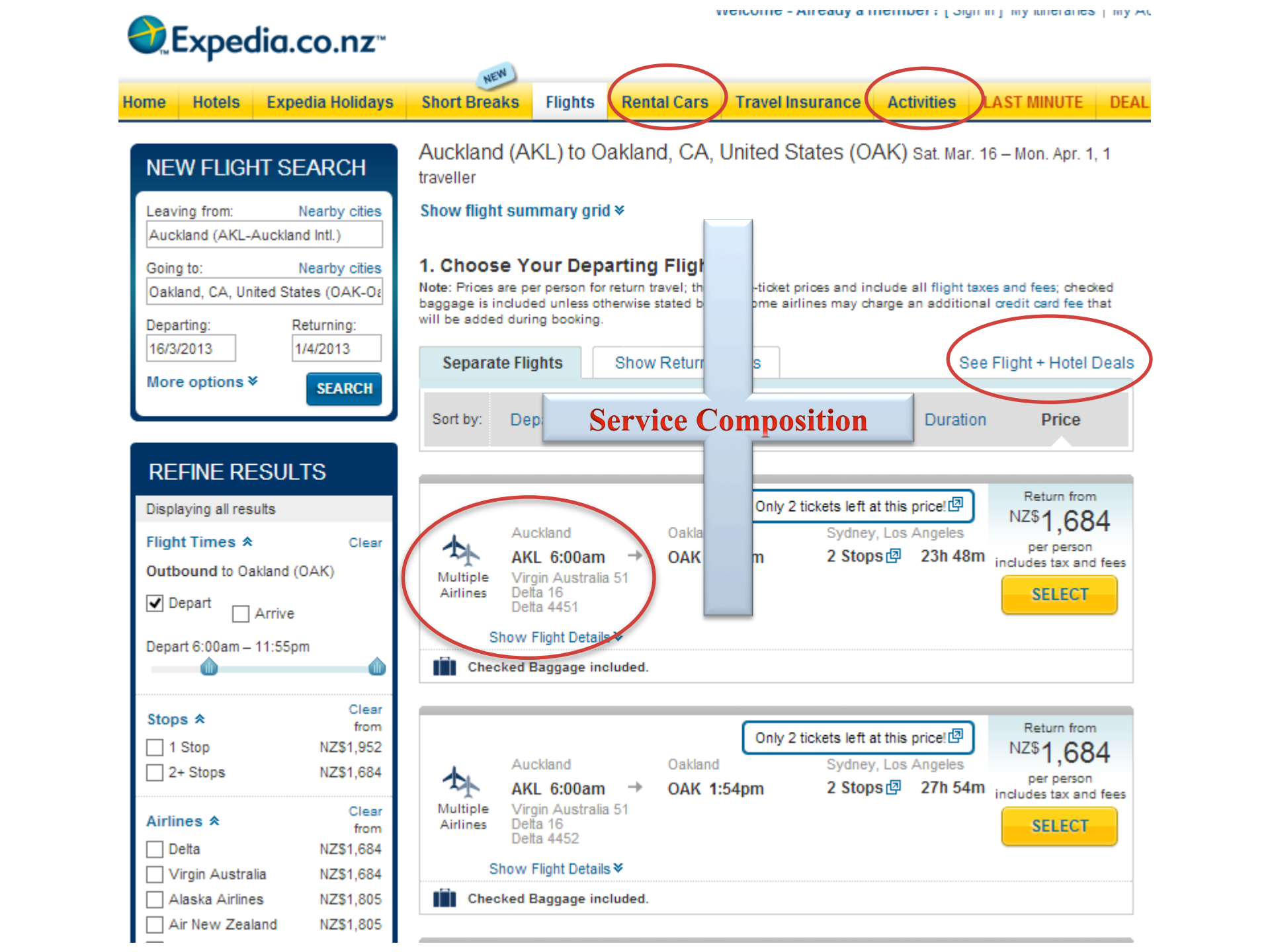
Timing/Functionality requirements

[Paolieri et al 2011] Towards Functional-Safe Timing-Dependable Real-Time Architectures.

Service Oriented Computing

Internet Service Composition

Featuring The Future …!


Expedia.co.nz
Home
Hotels
Expedia Holidays
NEW
Short Breaks
Flights
Rental Cars
Travel Insurance
Activities
LAST MINUTE
DEAL
NEW FLIGHT SEARCH
Leaving from:
Nearby cities
Auckland (AKL-Auckland Intl.)
Going to:
Nearby cities
Oakland, CA, United States (OAK-Oa
Departing:
16/3/2013
Returning:
1/4/2013
More options
SEARCH
REFINE RESULTS
Displaying all results
Flight Times
Clear
Outbound to Oakland (OAK)
Depart
Arrive
Depart 6:00am – 11:55pm
Stops
Clear
from
1 Stop
NZ$1,952
2+ Stops
NZ$1,684
Airlines
Clear
from
Delta
NZ$1,684
Virgin Australia
NZ$1,684
Alaska Airlines
NZ$1,805
Air New Zealand
NZ$1,805
Auckland (AKL) to Oakland, CA, United States (OAK) Sat. Mar. 16 – Mon. Apr. 1, 1 traveller
Show flight summary grid
1. Choose Your Departing Flig
Note: Prices are per person for return travel; th -ticket prices and include all flight taxes and fees; checked baggage is included unless otherwise stated b ome airlines may charge an additional credit card fee that will be added during booking.
Separate Flights
Show Retur s
See Flight + Hotel Deals
Sort by:
Dep
Service Composition
Duration
Price
Only 2 tickets left at this price!
Return from
NZ$1,684
per person
includes tax and fees
SELECT
Multiple Airlines
Auckland
AKL 6:00am
Virgin Australia 51
Delta 16
Delta 4451
Oakla
OAK m
Sydney, Los Angeles
2 Stops
23h 48m
Show Flight Details
Checked Baggage included.
Only 2 tickets left at this price!
Return from
NZ$1,684
per person
includes tax and fees
SELECT
Multiple Airlines
Auckland
AKL 6:00am
Virgin Australia 51
Delta 16
Delta 4452
Oakland
OAK 1:54pm
Sydney, Los Angeles
2 Stops
27h 54m
Show Flight Details
Checked Baggage included.

Related work

- Abstract Interfaces [Parnas'77]
- OO methodologies and UML
- Formal techniques:
  - IO Automata
  - Interface Automata
  - Interface Theories
  - Discrete controller synthesis
  - Module checking
  - Converter synthesis

Two key questions

- Question 1: specification matching / component adaptation (the “what” question).
- Question 2: component composition (the “how” question).

First Question:

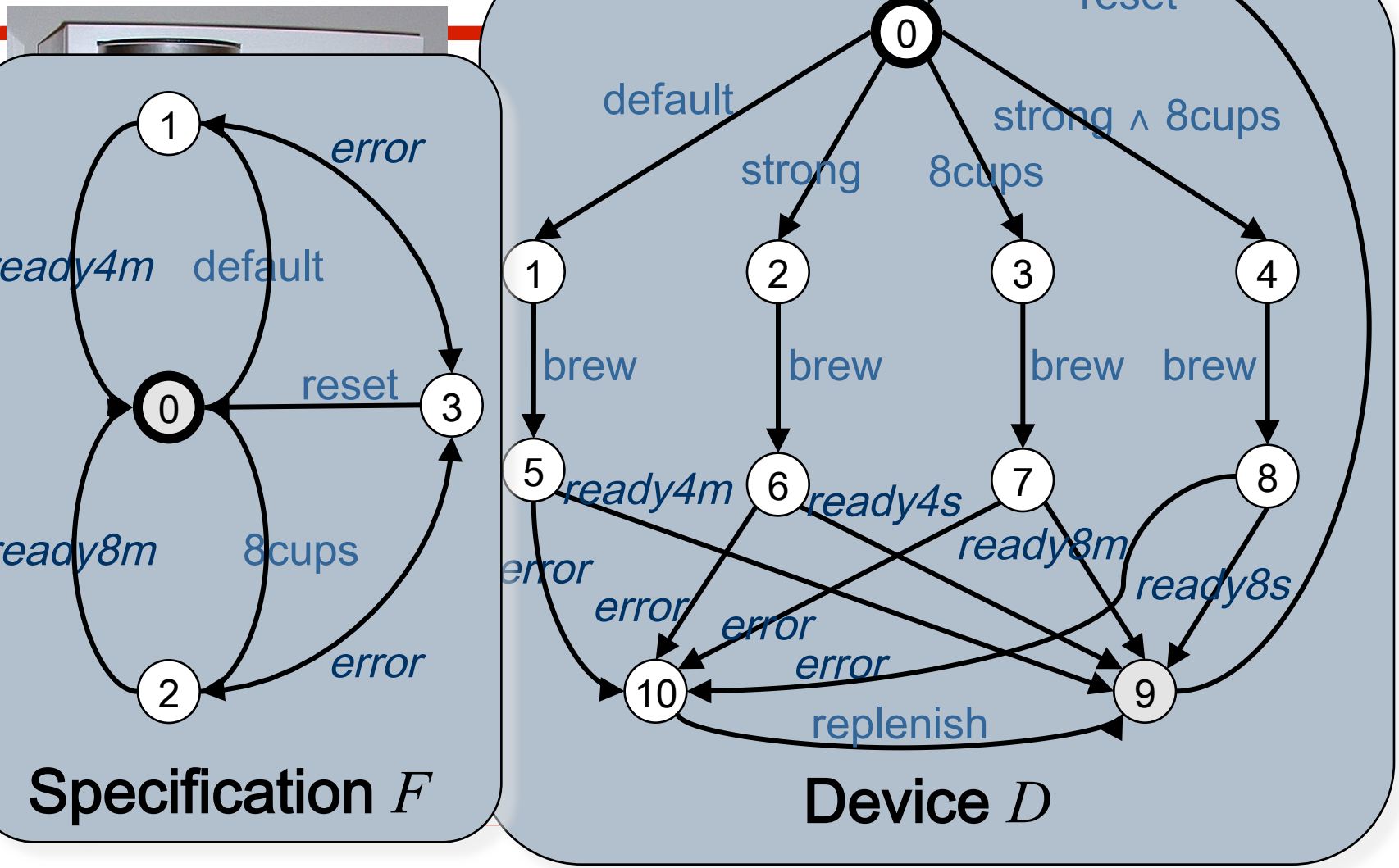
Specification Matching –

"Can a given device automatically be adapted to implement a new function?"

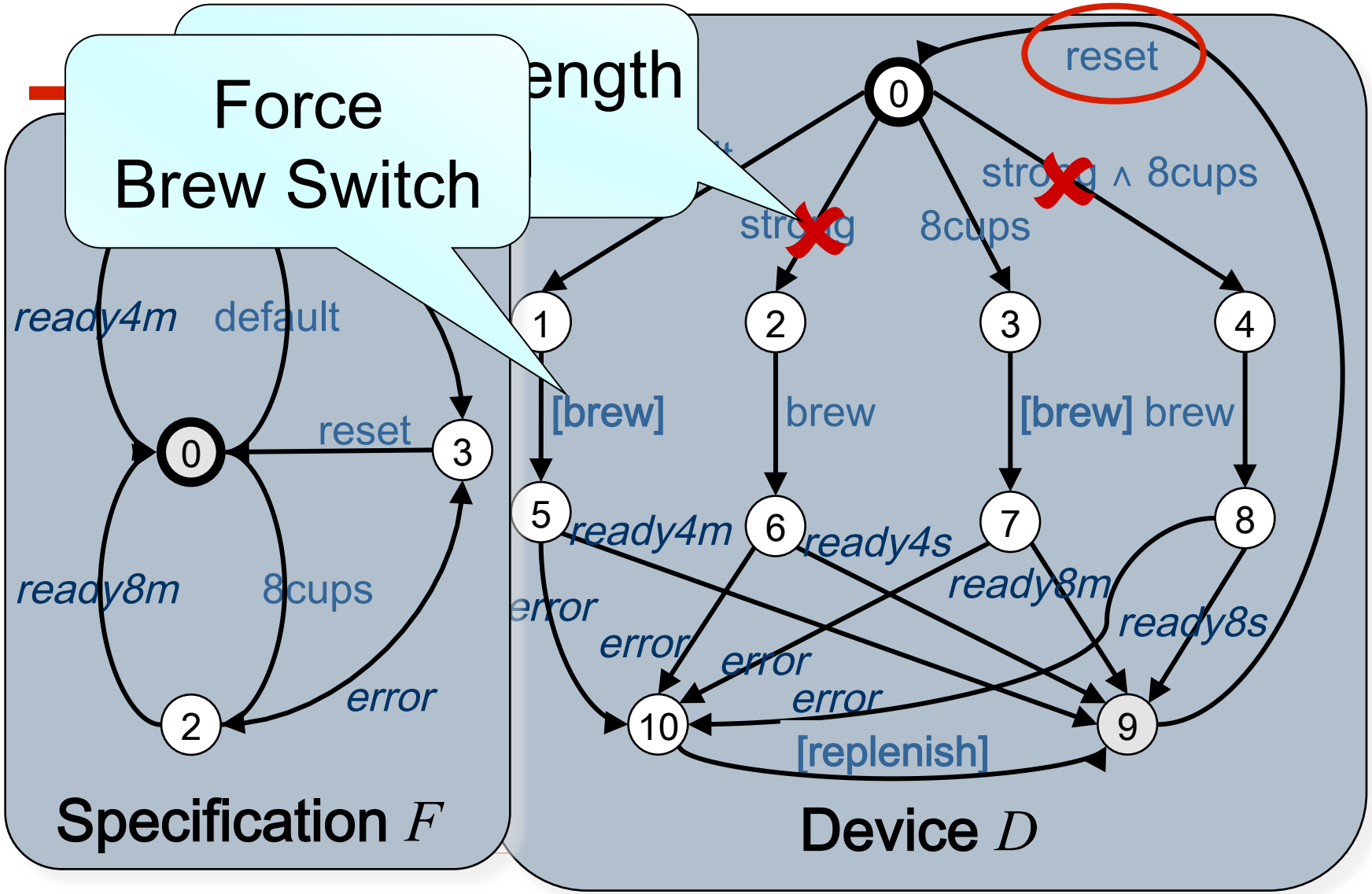
Two Answers:

- **Forced Simulation**
- **Supervisory Control**

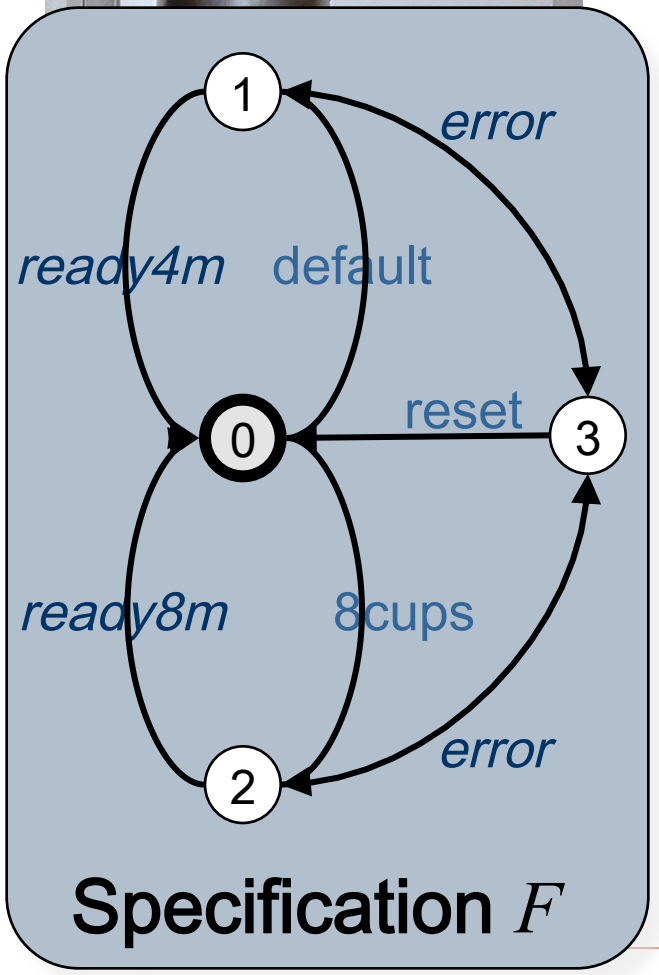
Coffee Brewer Example

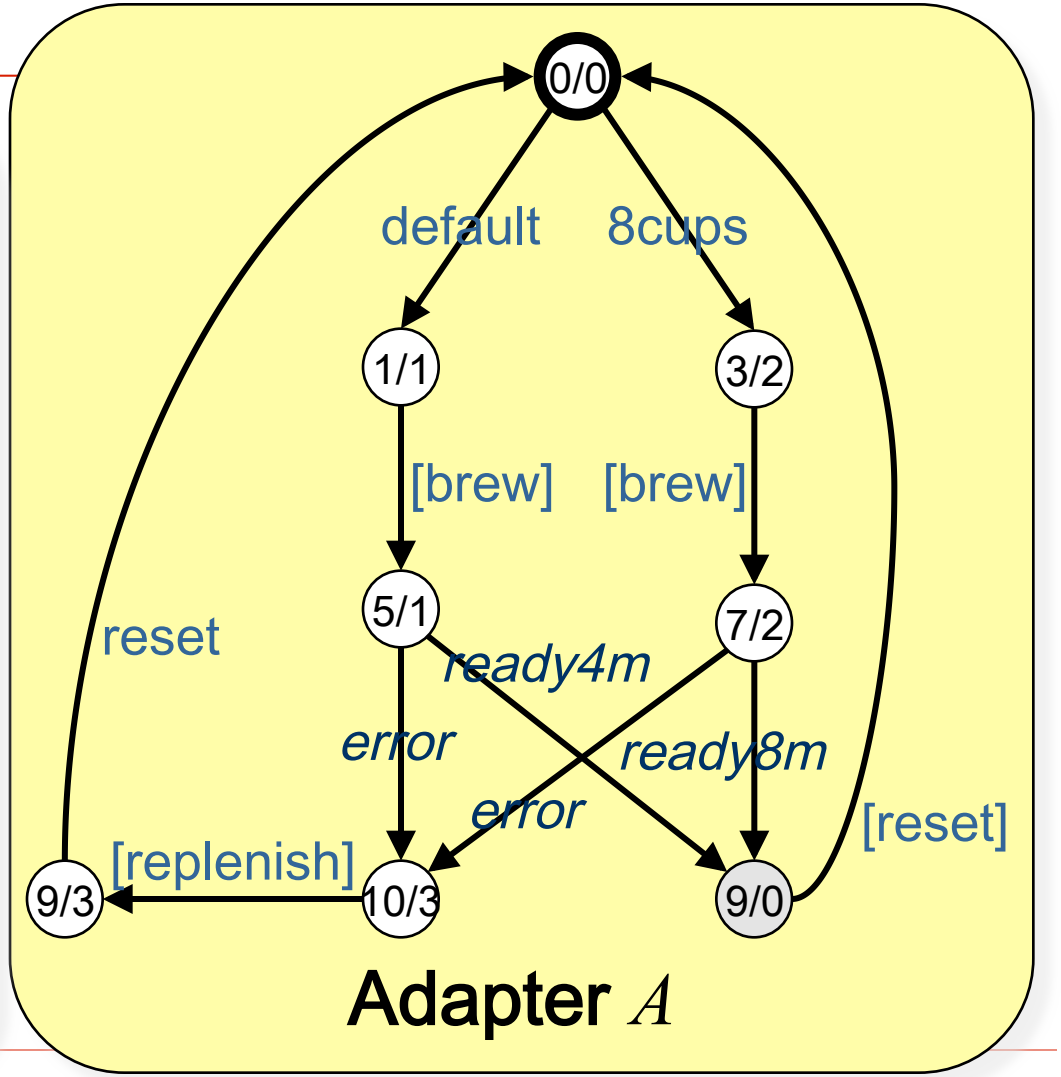


Disabling and Forcing



An Adapter for the Coffee Brewer





Forced Composition

Let A be an adapter an D be a device. Define the **forced composition** $A \,//\, D$ by

$$\frac{(q_A) \xrightarrow{[\alpha]}_A (q_A') \quad (q_D) \xrightarrow{\alpha}_D (q_D')}{(q_A, q_D) \xrightarrow{\tau} (q_A', q_D')}$$

$$\frac{(q_A) \xrightarrow{\sigma}_A (q_A') \quad (q_D) \xrightarrow{\sigma}_D (q_D')}{(q_A, q_D) \xrightarrow{\sigma} (q_A', q_D')}$$

Specification Matching Problem

Let F be a specification and D be a device. We say that

"D can implement the function F",

if there exists a well-formed and deterministic adapter A such that

$$A \,//\, D \approx F$$

Forced Simulation Solution

Theorem

There exists a well-formed and deterministic adapter A such that

$$A \,//\, D \approx F$$

if and only if

$$F \lesssim_{\text{fsim}} D$$

Condition for the existence of *A*

$R \subseteq Q_F \times Q_D \times \Sigma^*$ is a forced simulation relation between F and D *provided* :

1. $q_F^0 R^s q_D^0$ for some $s \in \Sigma^*$;

2. If $q_F R^{\sigma.s} q_D$ for $\sigma \in \Sigma$ and $s \in \Sigma^*$, then there exists $q'_D \in Q_D$ such that $q_D \xrightarrow{\sigma} q'_D$ and $q_F R^s q'_D$;

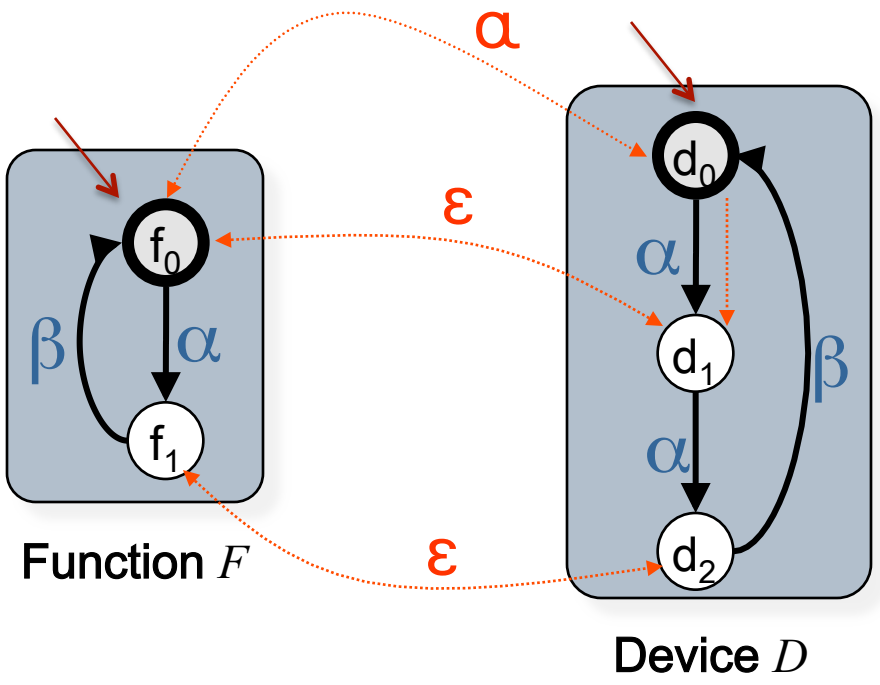
3. If $q_F R^{\varepsilon} q_D$ for all $\sigma \in \Sigma$ and all $q'_F \in Q_F$ such that $q_F \xrightarrow{\sigma} q'_F$, there exits $q'_D \in Q_D$ and $s \in \Sigma^*$ such that $q_D \xrightarrow{\sigma} q'_D$ and $q'_F R^s q'_D$.

Start states must be related

states related by a forcing sequence

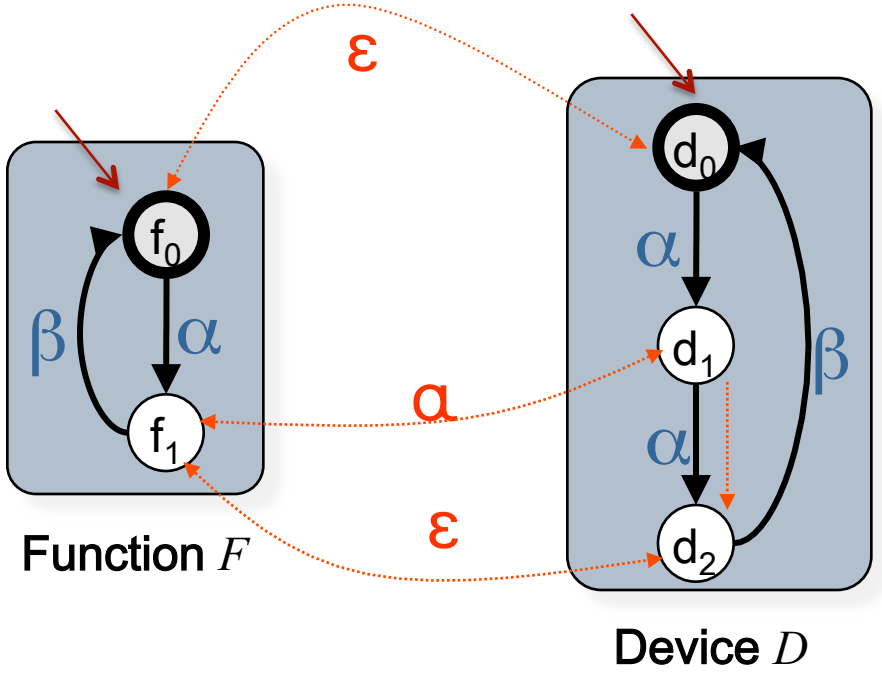
Directly related

Example



$R = \{(f_0, d_0, \alpha), (f_0, d_1, \varepsilon), (f_2, d_2, \varepsilon)\}$

Another Solution



R = {(f_0, d_0, ε), (f_0, d_1, α), (f_2, d_2, ε)}

Supervisory Control Problem

Let F be a specification and P be a plant. We say that

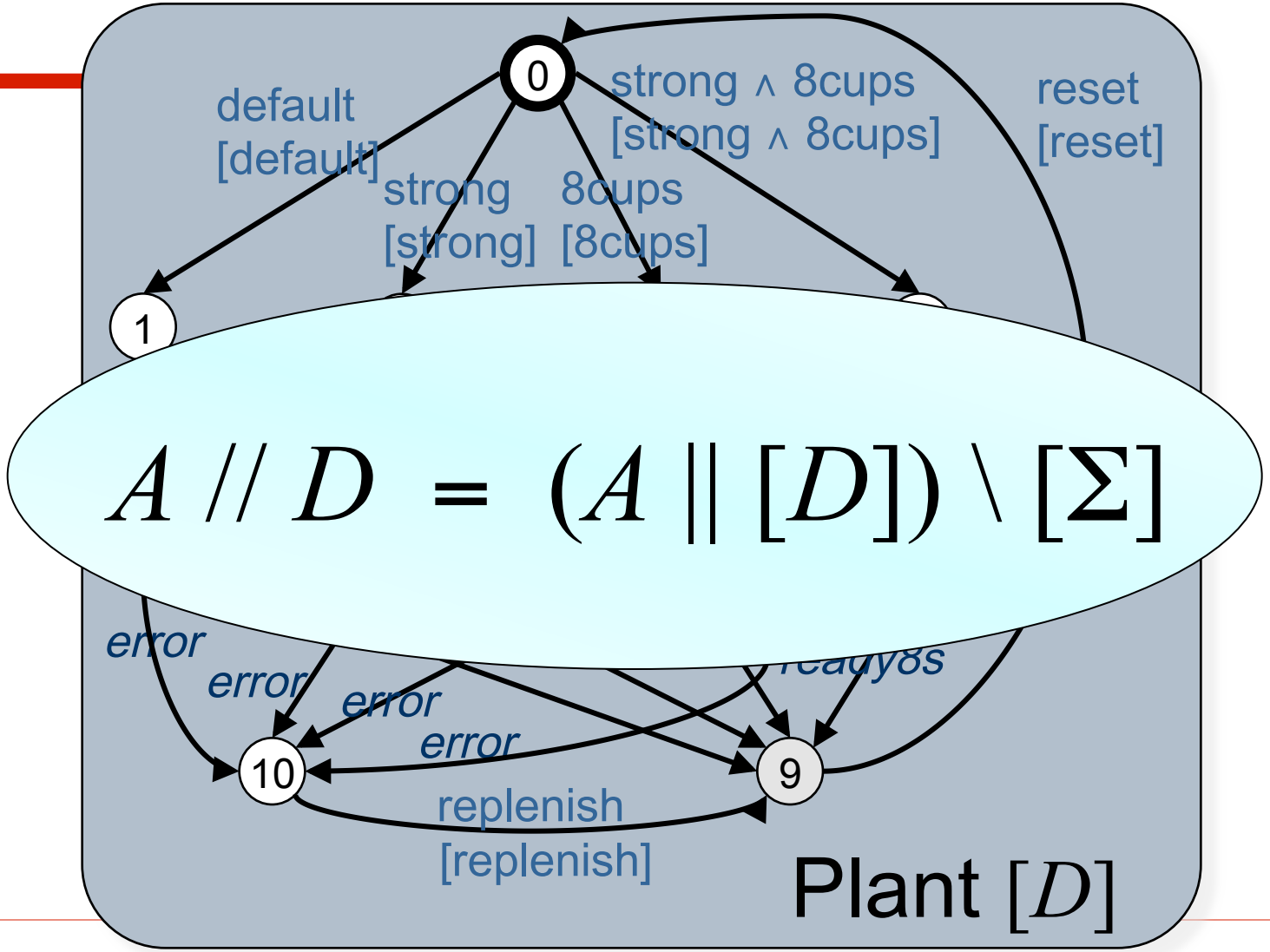
"F can be achieved by control of P"

"F is controllable with respect to P"

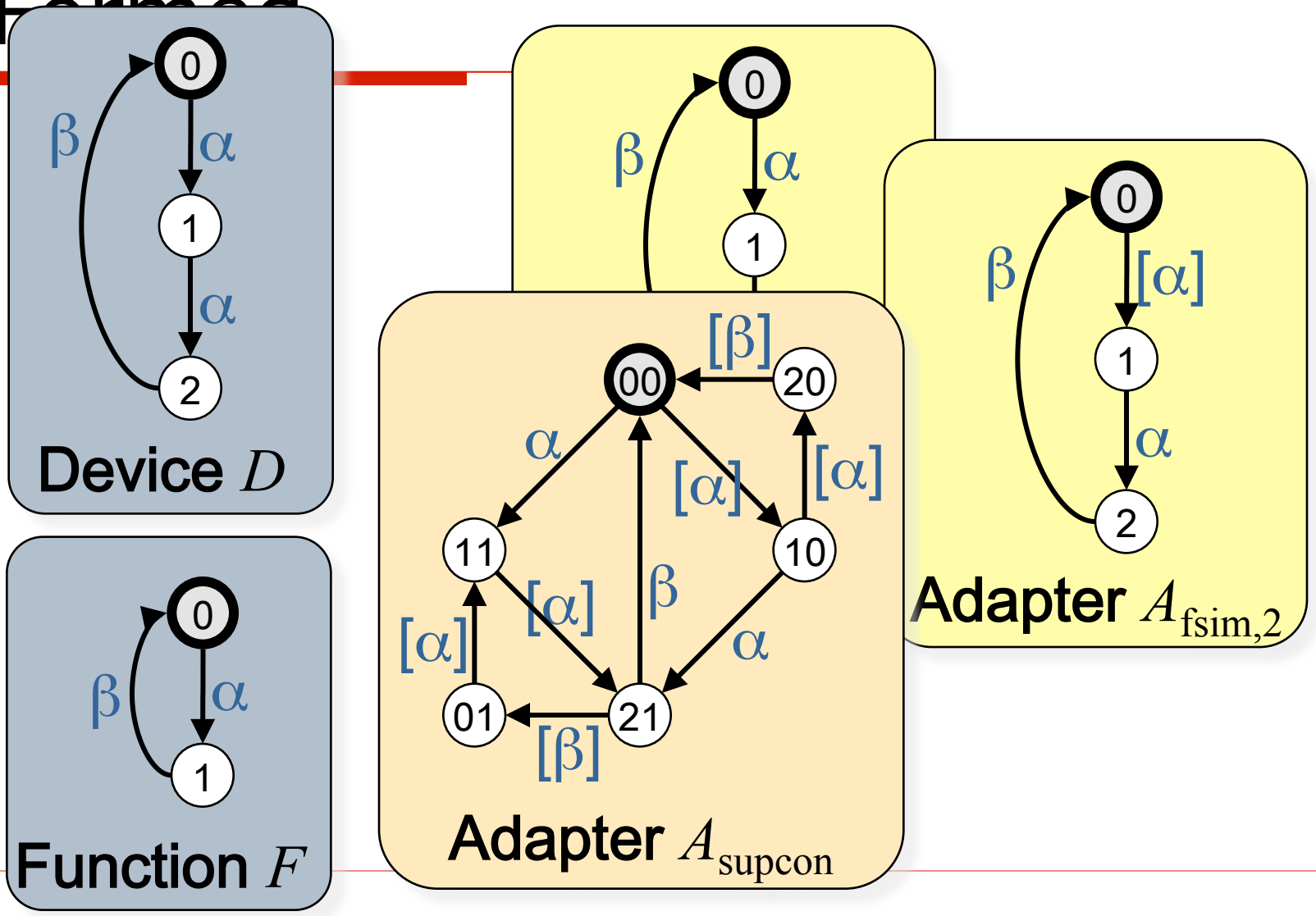
if there exists a **supervisor** S such that

$$\boldsymbol{L}(S \,||\, P) = \boldsymbol{L}(F)$$

Creating a Plant from the Device



Least Restrictive vs. Well-Formed



Comparison and Summary

Feature	**Forced simulation**	**Supervisory control**
Relationship between A and F	$A \,//\, D \approx F$	$\mathbf{L}(A \,//\, D) \subseteq \mathbf{L}(F)$
Well-formedness	guaranteed	requires additional steps
Forced cycles	not possible	may occur
Nonblocking	guaranteed	can be guaranteed
Uniqueness	solutions weakly bisimilar	unique least restrictive solution
Controllability	not considered	handled
Complexity	$\boldsymbol{O}(\lvert Q_F\rvert\lvert Q_D\rvert^2\lvert\Sigma\rvert)$	$\boldsymbol{O}(\lvert Q_F\rvert^2\lvert Q_D\rvert^2\lvert\Sigma\rvert)$

Second Question:

- **Composition –**Design and develop systems from multiple independently developed components
- *How to effectively address protocol-mismatches during composition?*

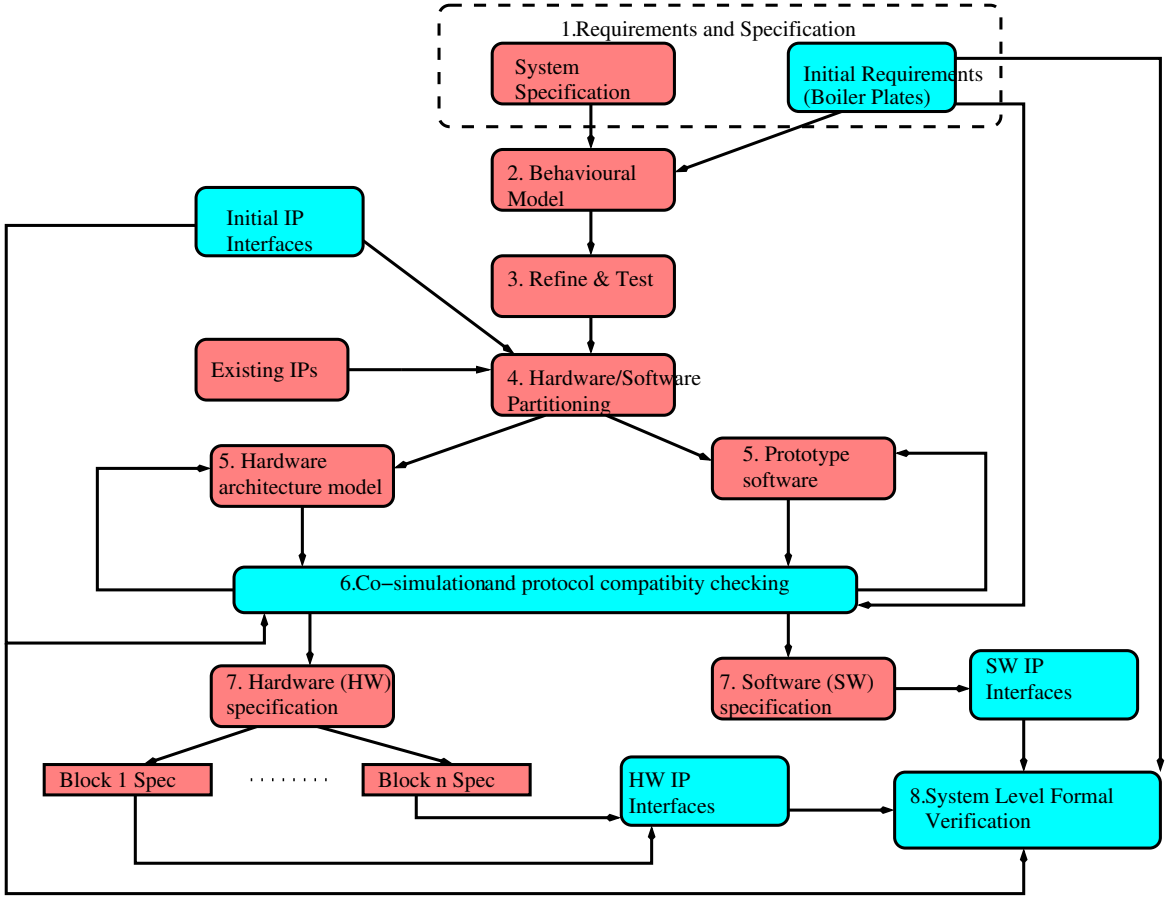
Answer:

Relationship to convertibility verification.

Motivation

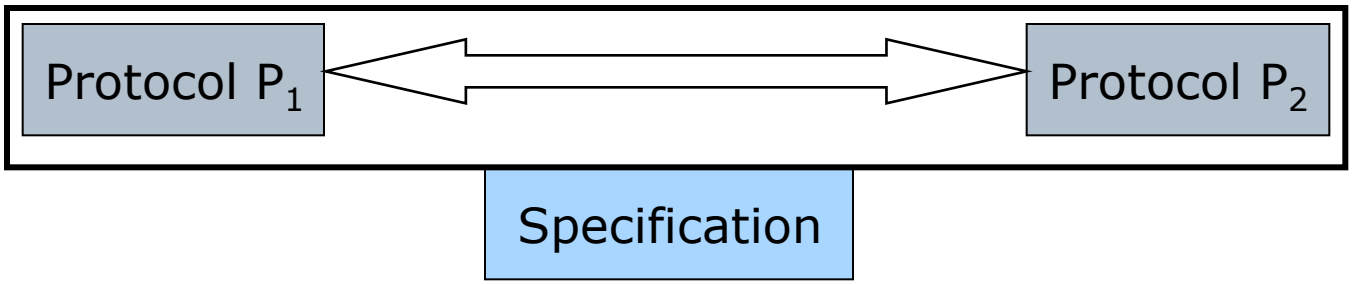
- **Reuse Methodology Manual for System-on-a-Chip Designs by Keating and Bricaud, Springer 2002 (3rd edition)**
  - "verifying functionality and timing at the system-level is probably the most difficult and important aspect of SoC design. .. For many teams, verification takes 50%-80% of the overall design effort"
  - "the low-level interfaces do not work; for example, a handshake signal inverted"

Suggested design flow



Solution Mechanism

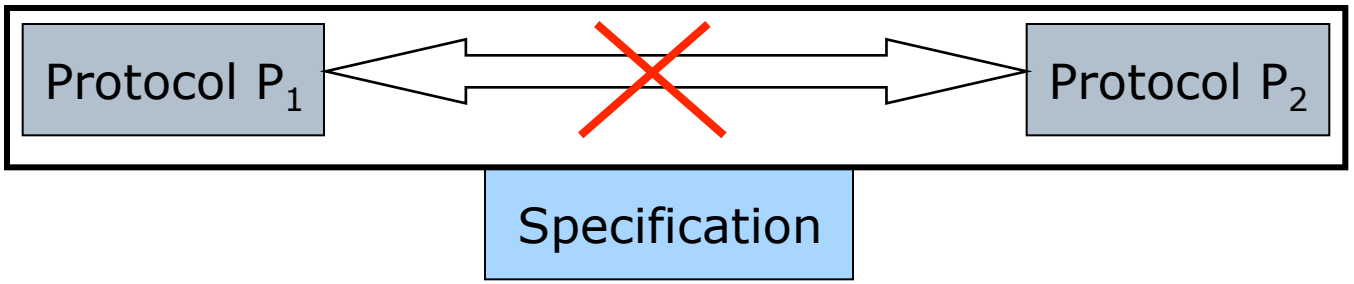
- Converter-based protocol conversion
  - Develop a converter: acts as a mediator between two components with mismatched protocols



Goal: Compose P_1 and P_2 to realize the Specification

Solution Mechanism

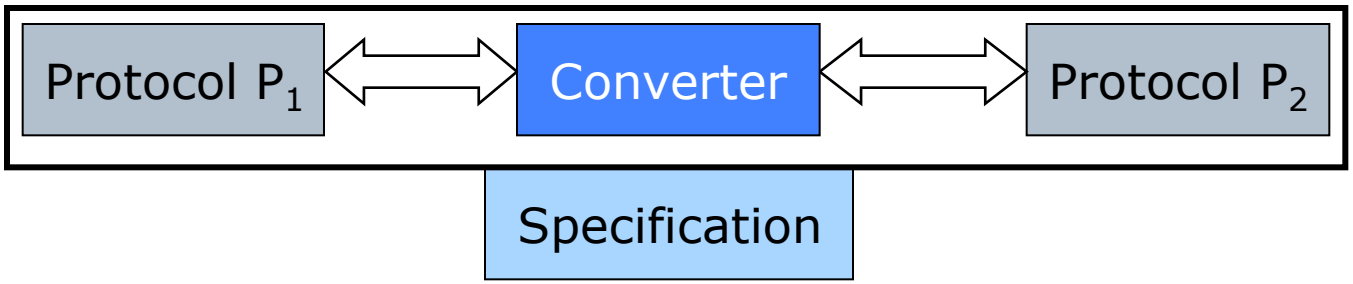
- Converter-based protocol conversion
  - Develop a converter: acts as a mediator between two components with mismatched protocols



Goal: Compose P_1 and P_2 to realize the Specification

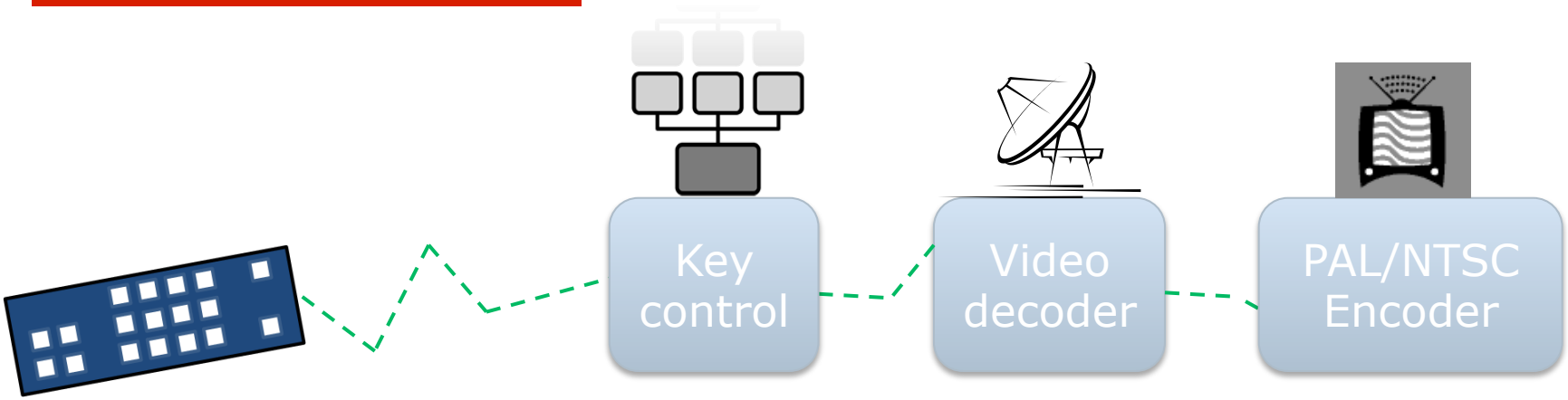
Solution Mechanism

- Converter-based protocol conversion
  - Develop a converter: acts as a mediator between two components with mismatched protocols



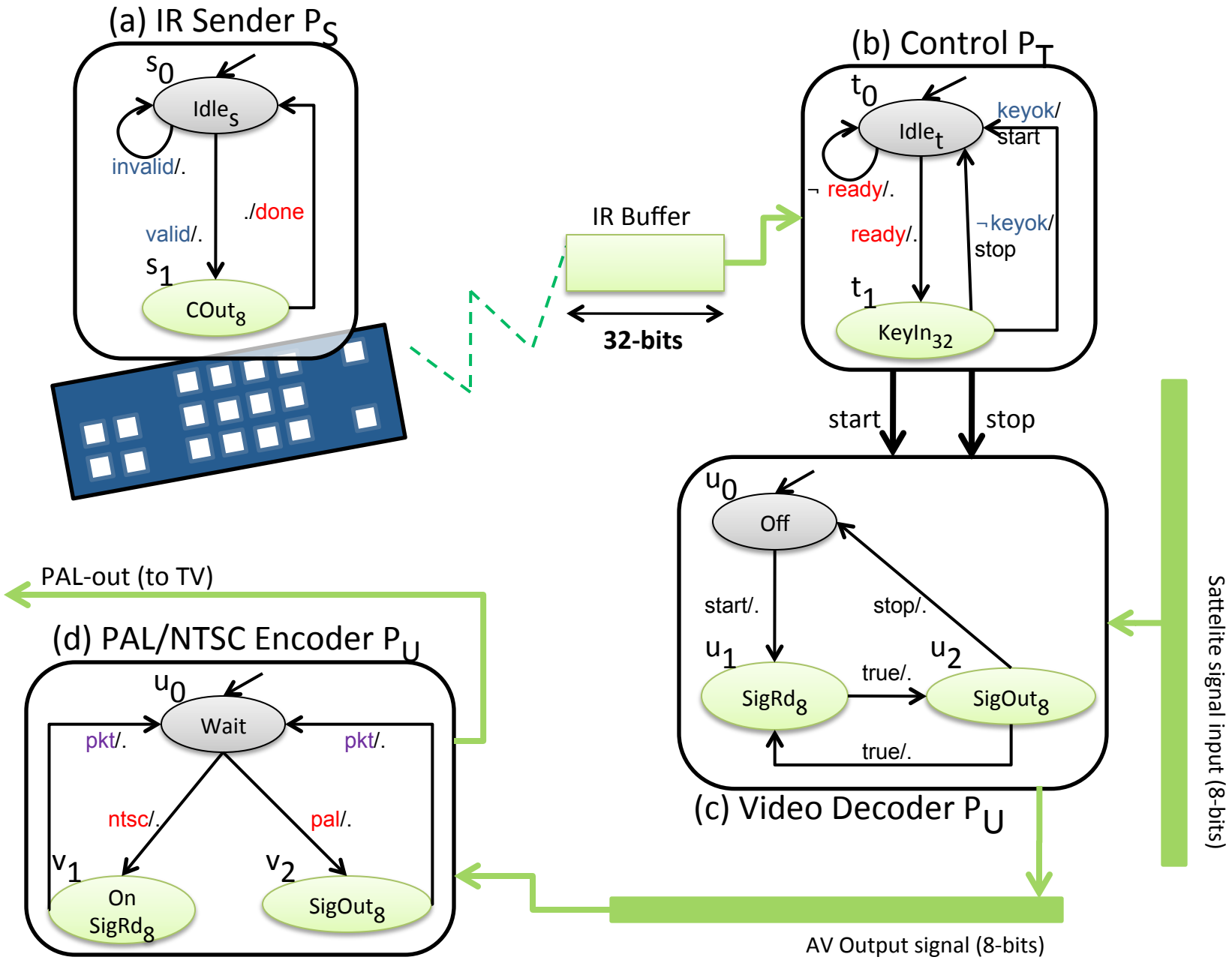
Solution: Converter addresses mismatches

Set-top box

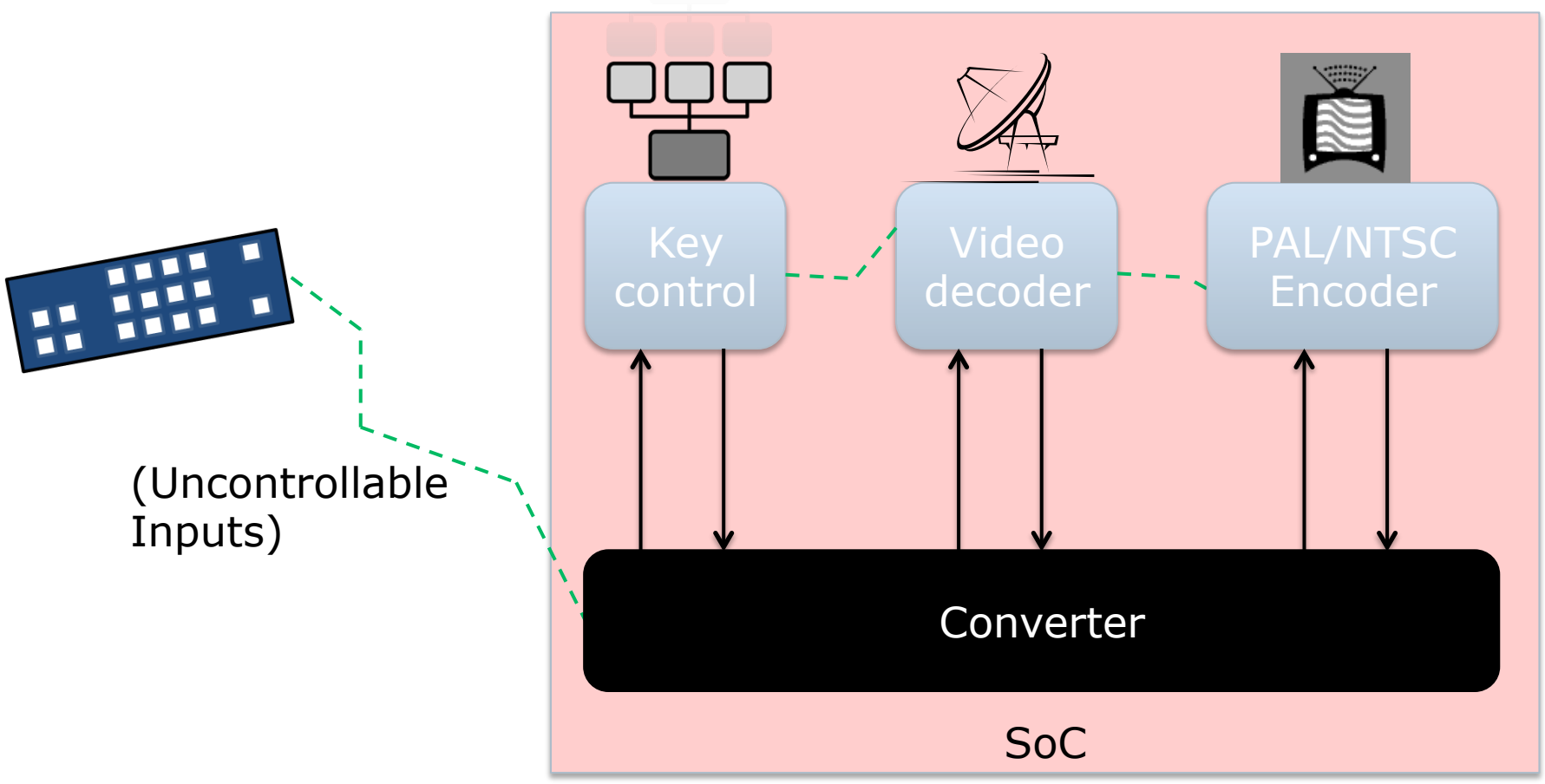


Challenges:

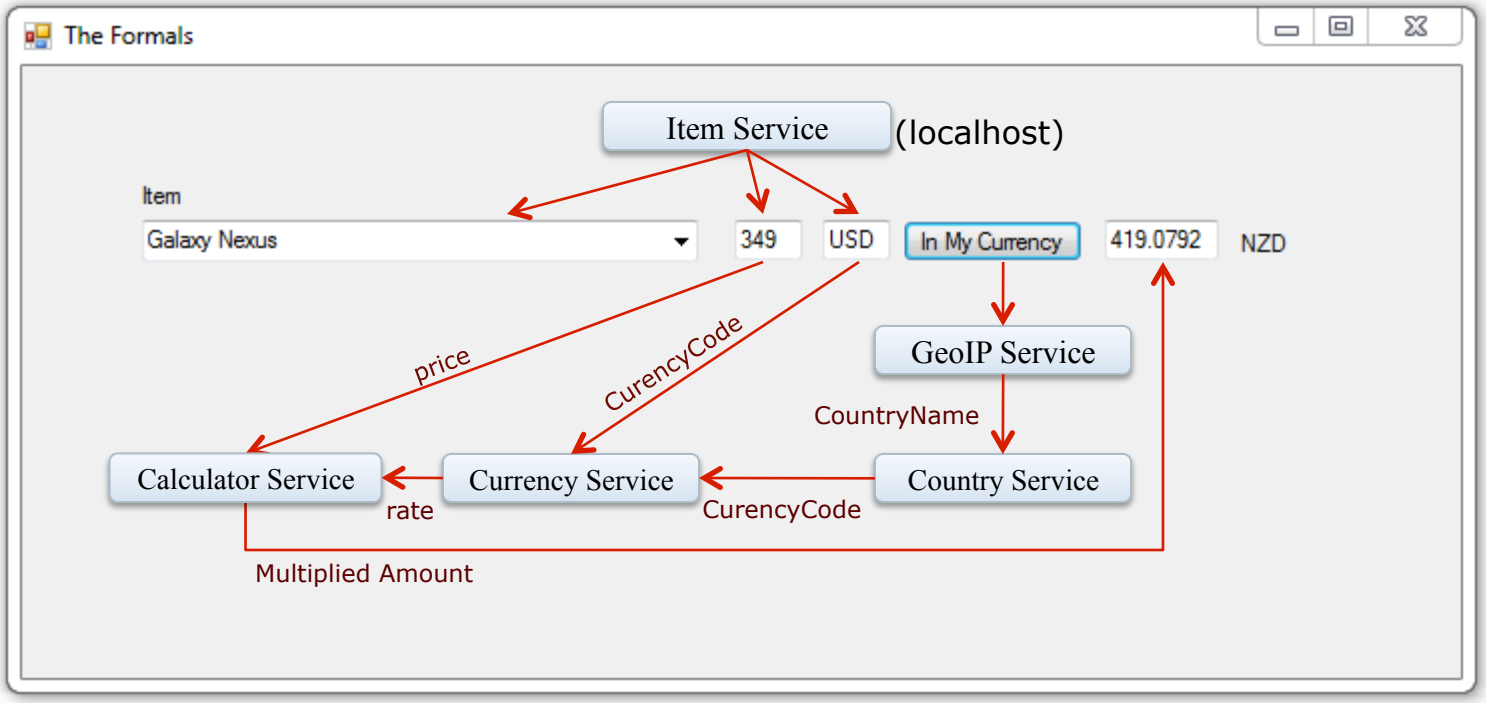
- Multi-clock
- Differing data-widths
- Control signals mismatch


(a) IR Sender P_S
s_0
$Idle_s$
invalid/.
./done
valid/.
s_1
$COut_8$
IR Buffer
32-bits
(b) Control P_T
t_0
$Idle_t$
keyok/
start
¬ ready/.
ready/.
¬keyok/
stop
t_1
$KeyIn_{32}$
start
stop
u_0
Off
start/.
stop/.
u_1
u_2
$SigRd_8$
true/.
$SigOut_8$
true/.
(c) Video Decoder P_U
Sattelite signal input (8-bits)
PAL-out (to TV)
(d) PAL/NTSC Encoder P_U
u_0
Wait
pkt/.
pkt/.
ntsc/.
pal/.
v_1
v_2
On $SigRd_8$
$SigOut_8$
AV Output signal (8-bits)

Converter

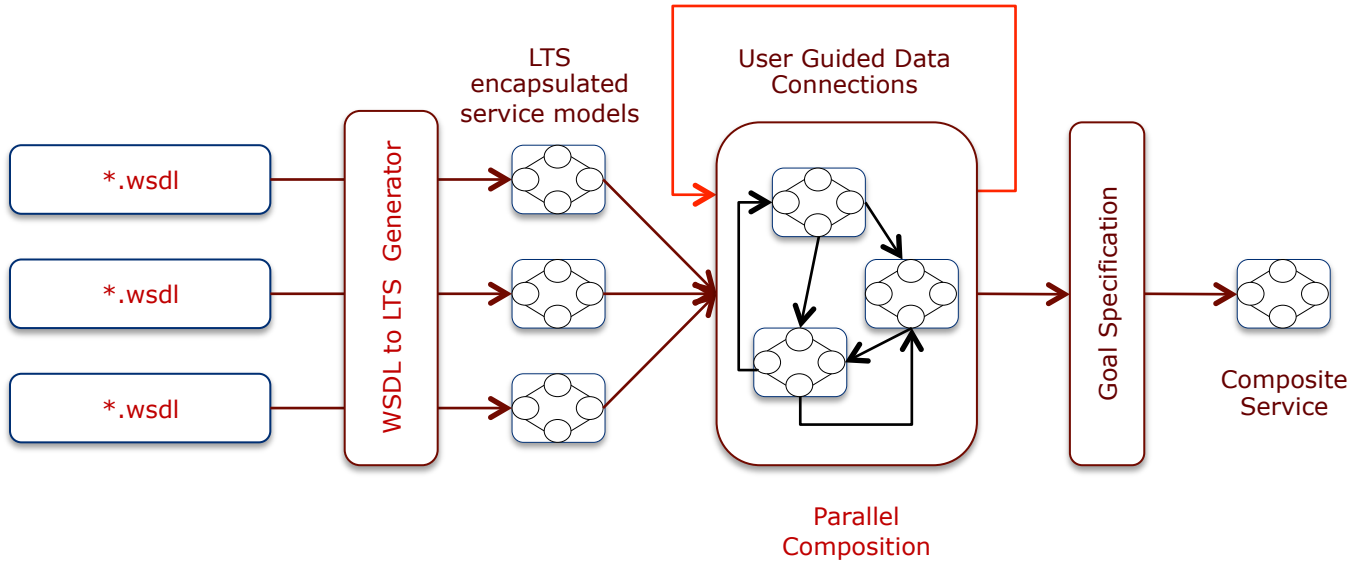

Key control
Video decoder
PAL/NTSC Encoder
(Uncontrollable Inputs)
Converter
SoC

How about service composition?



Click for Demo

Composition Framework



Related Work

Approach	Model	Spec	Multiple Protocols	Algorithm	UE	Buffering	Data	Multi-clock
Avnit et al.'08	SPA	nil	no	refinement	no	yes	limited	yes
D'Silva et al.'04	SPA	Nil	no	Refinement	no	yes	limited	yes
Passerone et al.'02	LTS	LTS	no	Game-theoretic	no	yes	no	no
Kumar et al.'97	LTS	LTS	no	Supervisory Control	yes	no	no	no
Tivoli et al.'08	LTS	Nil	Yes	Coverability-analysis	yes	yes	no	yes
Our Approach	LTS	CTL	yes	Model checking	yes	yes	yes	yes

Related Work – Service Composition

Approach	Input Services		Data				Behaviour		Composition Algorithm
	Type	Auto Model	Model	Flow	Sem-antic	Struc-tural	Model	Spec	
Mitra et al.	Syntactic	-	-	-	-	-	i/o automata	i/o automaton	Tabled-Logic Programming
ASTRO	Syntactic	-	DataNet	+	-	-	STS	EAGLE	Planning based Model Checking
Berardi et al.	Syntactic	-	-	-	-	-	FSM	DPDL	Satisfiability of DPDL
Lecue et al.	Semantic	-	Schema Graph	+	+	+	-	-	-
Proposed	Syntactic	+	Schema Graph	+	+	+	LTS	CTL	Tableau based Model Checking

Types of Protocol Mismatch

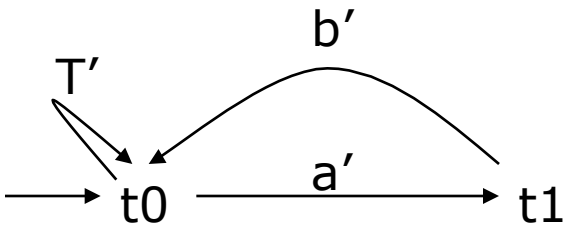
- **Control-signal mismatch**
- Data mismatch
- Clock mismatch

Ref: **Convertibility Verification and Converter Synthesis: Two Faces of the Same Coin**

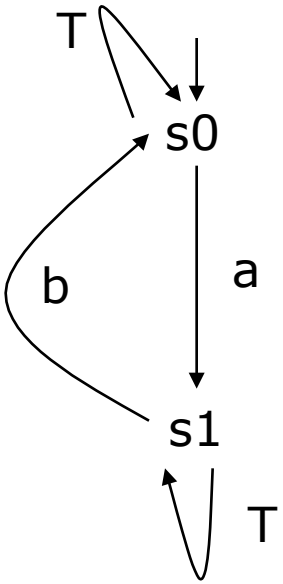
Passerone, Luca de Alfaro, Thomas A. Henzinger and Alberto L. Sangiovanni-Vincentelli, ICCAD'02

Protocol Model

primed: input signal
unprimed: output signal



Serial Consumer



Handshake Producer

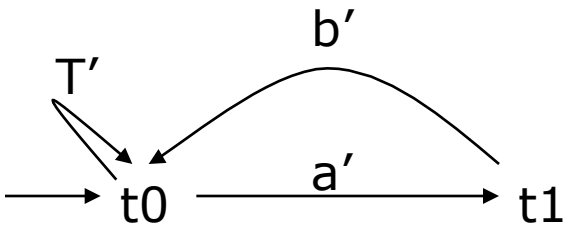
$KS = (AP, S, s_0, \Sigma, R, L)$
AP: atomic propositions, S: set of states,
$s_0 \in S$: start state, Σ: transition labels,
R: Transitions, L: labels states to propositions.

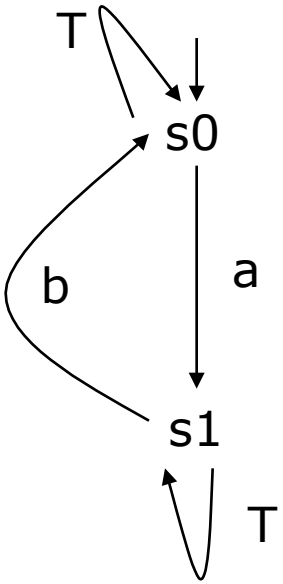
Ref: **Convertibility Verification and Converter Synthesis: Two Faces of the Same Coin**

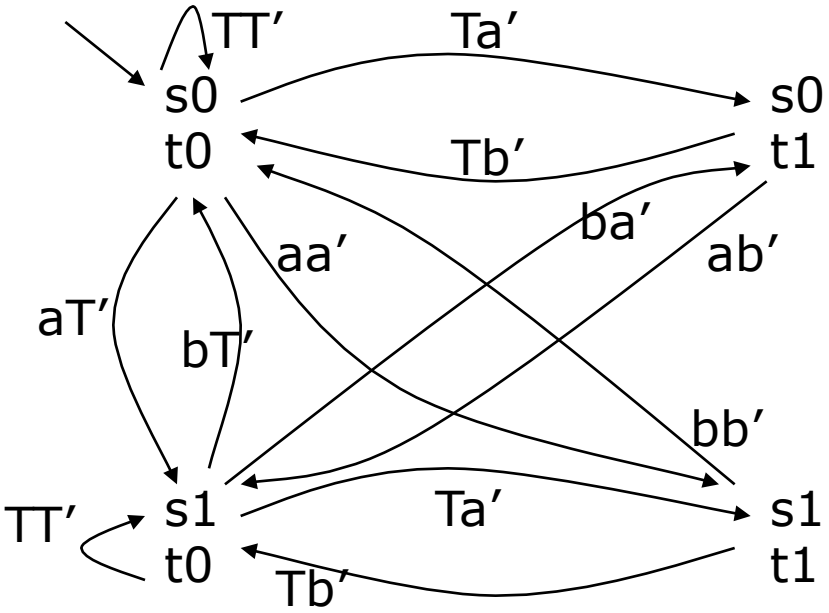
Passerone, Luca de Alfaro, Thomas A. Henzinger and Alberto L. Sangiovanni-Vincentelli, ICCAD'02

Protocol Model Composition

primed: input signal
unprimed: output signal







Specification Language

- CTL Syntax

$\Phi \rightarrow$ tt | P | $\neg$P | $\Phi \vee \Phi$

| AX/EXΦ — All/some successors satisfy Φ

| AG/EGΦ — All/some reachable states satisfy Φ

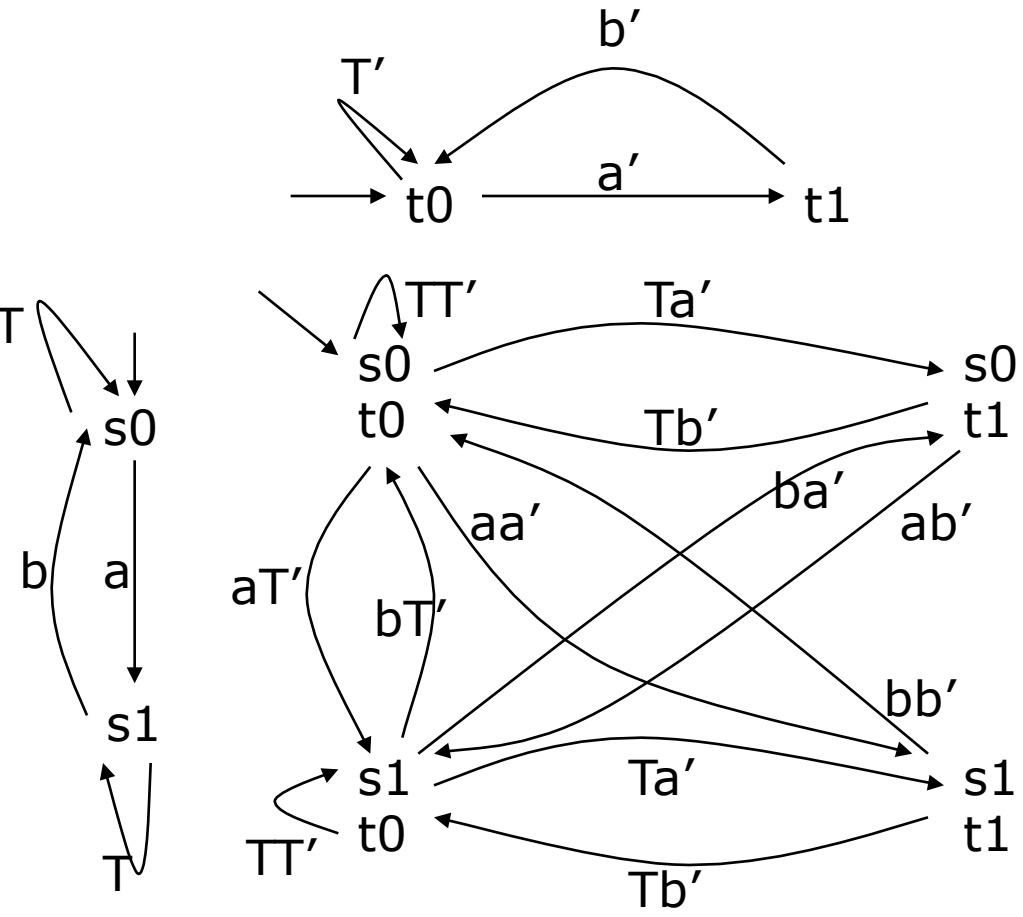
| A(Φ U Ψ) — Along all paths Φ is satisfied until Ψ

| E(Φ U Ψ) — Along some path Φ is satisfied until Ψ

Ref: **Convertibility Verification and Converter Synthesis: Two Faces of the Same Coin**

Passerone, Luca de Alfaro, Thomas A. Henzinger and Alberto L. Sangiovanni-Vincentelli, ICCAD'02

Protocol Model Properties



Input cannot be made before corresponding output:

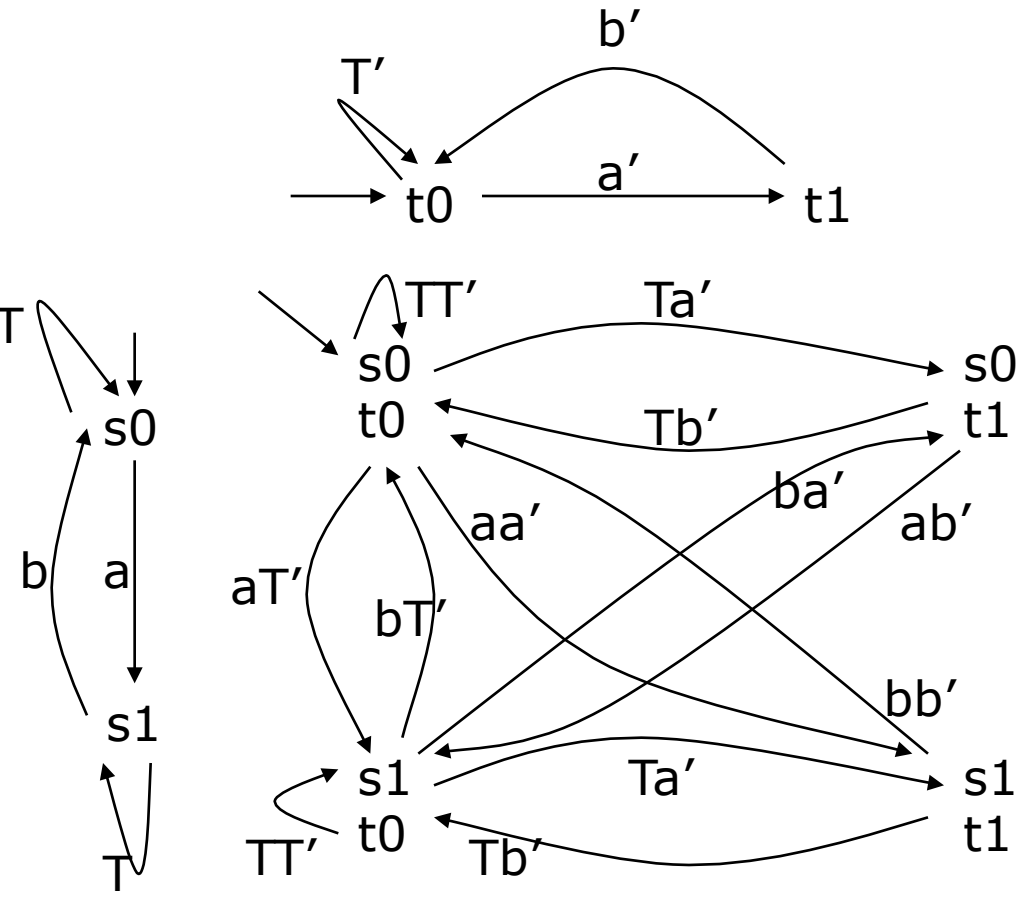
1. AG[s0,t0 ⇒ AX¬(¬s1,t1)]
2. AG[s1,t1 ⇒ AX¬(¬s0,t0)]

(s0,t0): for a action
(s1,t1): for b action

Ref: **Convertibility Verification and Converter Synthesis: Two Faces of the Same Coin**

Passerone, Luca de Alfaro, Thomas A. Henzinger and Alberto L. Sangiovanni-Vincentelli, ICCAD'02

Protocol Model Properties



Input cannot be made before corresponding output:

1. AG[s0,t0 ⇒ AX¬(¬s1,t1)]
2. AG[s1,t1 ⇒ AX¬(¬s0,t0)]

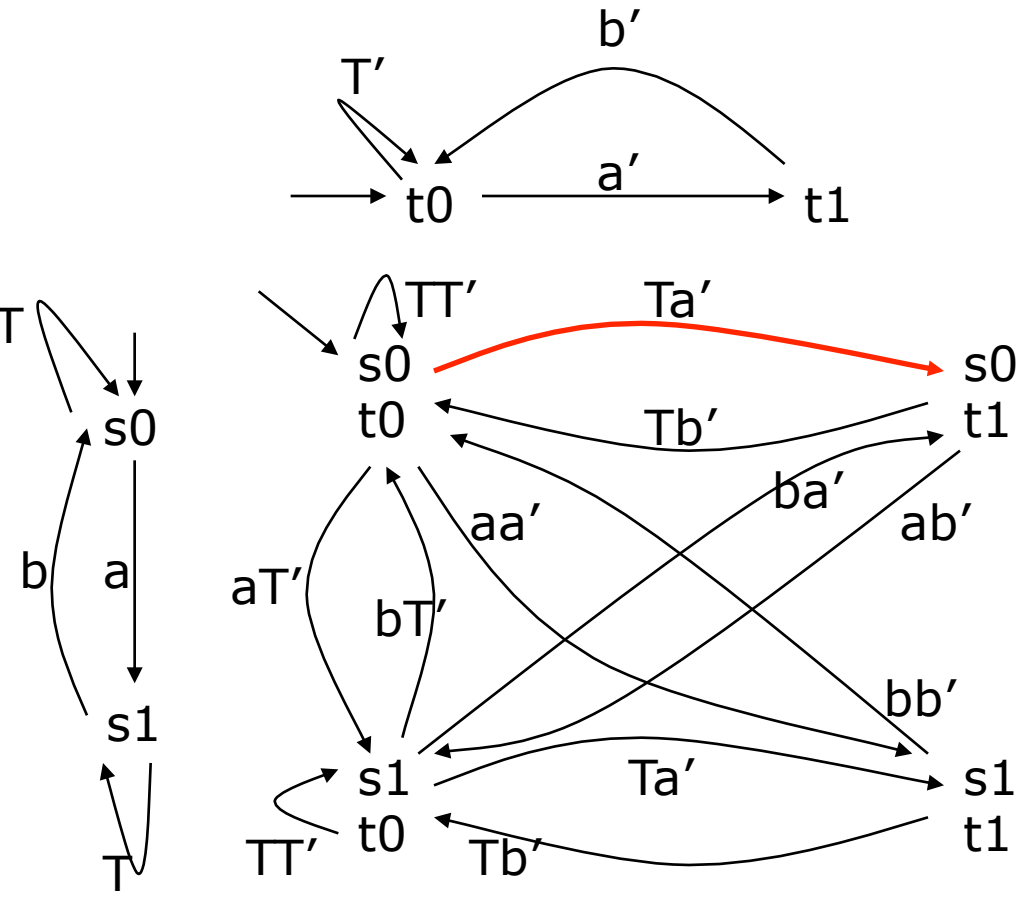
Output of b/a is not allowed before a/b is received:

1. AG[s1,t0 ⇒ AX¬(s0t0)]
2. AG[s0,t1 ⇒ AX¬(s1t1)]

Ref: **Convertibility Verification and Converter Synthesis: Two Faces of the Same Coin**

Passerone, Luca de Alfaro, Thomas A. Henzinger and Alberto L. Sangiovanni-Vincentelli, ICCAD'02

Protocol Model Properties



Input cannot be made before corresponding output:

1. AG[s0,t0 ⇒ AX¬(¬s1,t1)]
2. AG[s1,t1 ⇒ AX¬(¬s0,t0)]

Output of b/a is not allowed before a/b is received:

1. AG[s1,t0 ⇒ AX¬(s0t0)]
2. AG[s0,t1 ⇒ AX¬(s1t1)]

Lock-Step Composition

- Converter-based solution
  - Protocol-models move if and only if the converter allows that move
  - Converter cannot block any outputs

- Let c_i be composed with (s_i, t_i) then

  $(s_i,t_i) \xrightarrow{a} (s_j, t_j)$ is allowed if and only if c_i can move on (a')

Converter Synthesis

$$\frac{(s,t)//c \models \Phi}{(s_1,t_1)//c_1 \models \Phi_1 \quad (s_2,t_2)//c_2 \models \Phi_2 \ldots (s_k,t_k)//c_k \models \Phi_k}$$

- The antecedent holds if and only if the consequents hold
- Local, top-down approach similar to tableau-based CTL model checking

Tableau Rules

$$\frac{(s,t)//c \models \Psi}{\exists\, \pi \subseteq \Pi : \forall\, \sigma \in \pi :\ (s_\sigma, t_\sigma)//c_\sigma \models \Psi_{AX}}$$

$$\begin{cases} \Psi_{AX} = \{\Phi \mid AX\Phi \in \Psi\} \\ \Pi \quad = \{\sigma \mid (s,t) \xrightarrow{\sigma} (s_\sigma, t_\sigma)\} \\ c_\sigma :\ c \xrightarrow{\sigma'} c_\sigma \quad \text{and} \quad D(\sigma,\sigma') \end{cases}$$

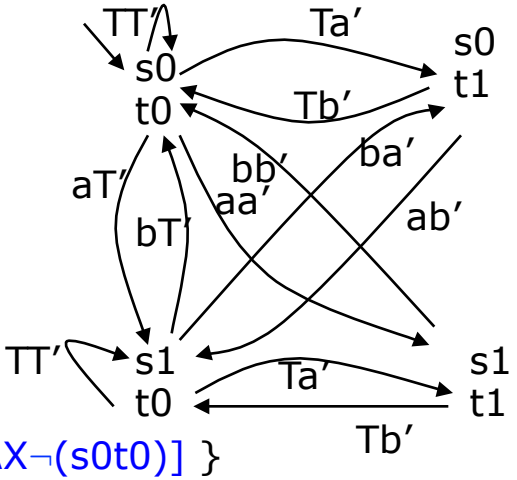
Ψ only contains formulas of the form $AX\Phi$

1. Identify the set of possible transitions from (s,t): Π
2. Enable a subset of possible transitions using converter: $c \rightarrow c_\sigma$
3. All enabled transition leads to states satisfying Φ's

Enabled transition set must include all possible output transitions. Also, resulting machine has to be responsive to T input.

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)

c0

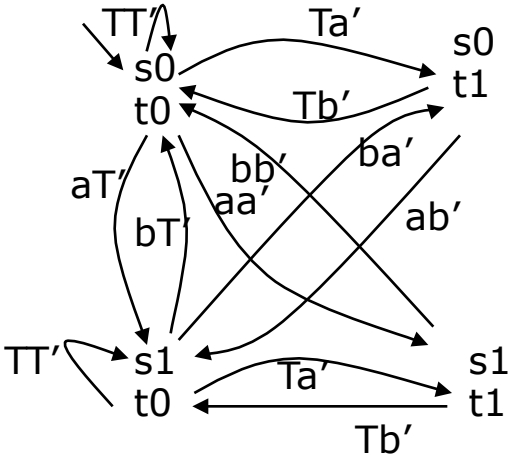


Example

s0t0//c0 ⊨ {AG[s0t0⇒AX¬(¬s1,t1)], AG[s1t1⇒AX¬(¬s0,t0)], AG[s1t0⇒AX¬(s0t0)] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)

c0

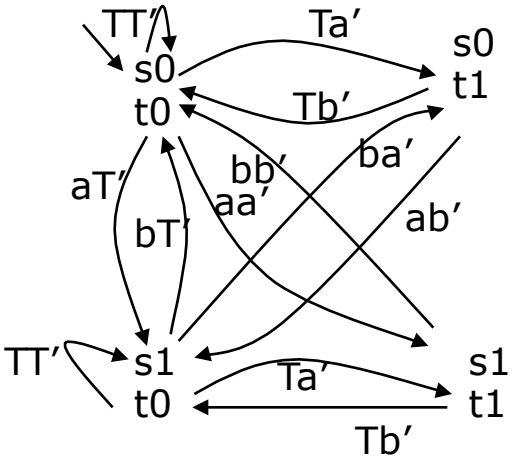


Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)

c0



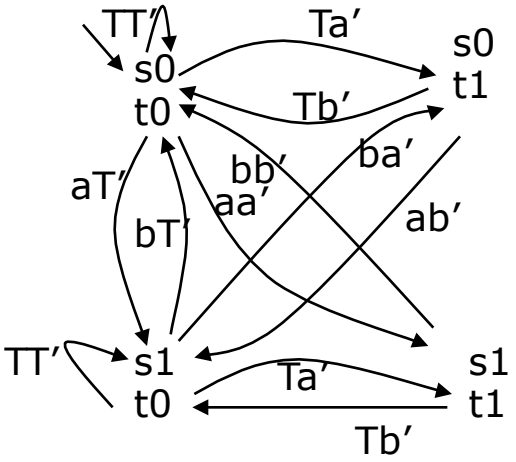
Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)

c0



Example

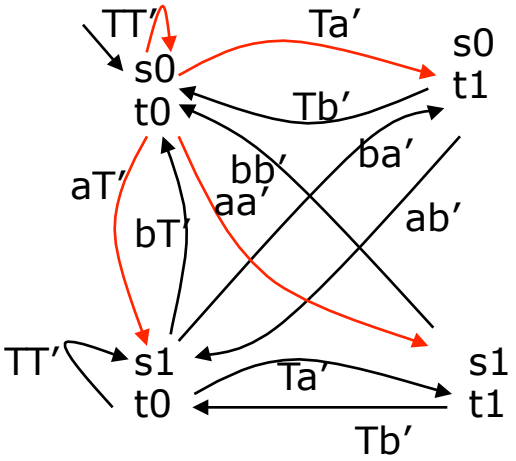
s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)

c0



Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

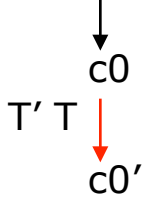
s0t0//c0′ ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

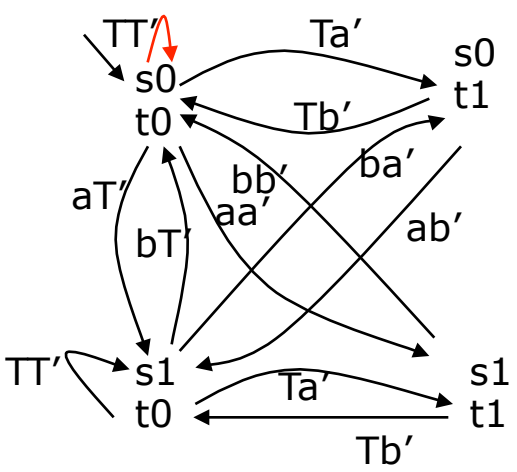
s0t1//c1 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s1t1//c2 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s1t0//c3 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

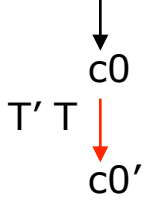
s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

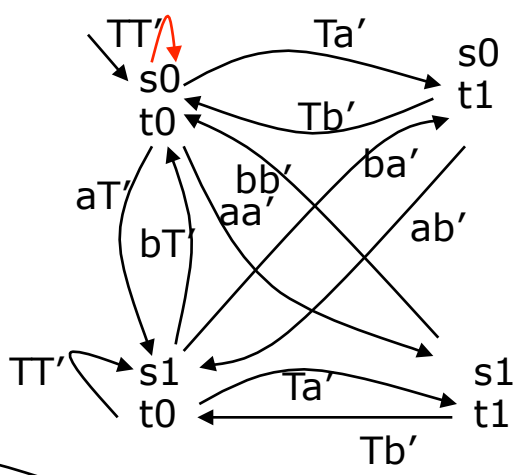
s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

s0t0//c0' ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

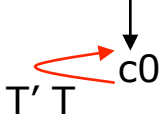
s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

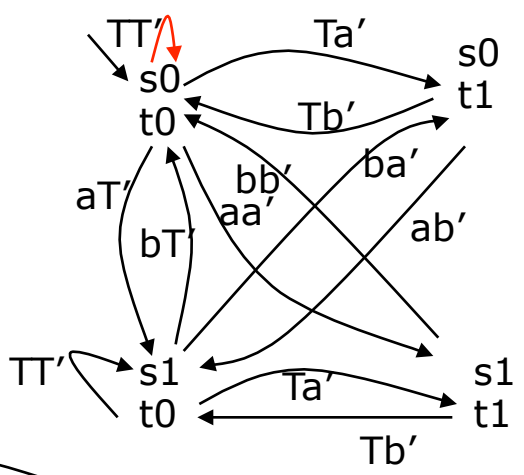
s0t0//c0' ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s0t0//c0' ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

Same formula state pair

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

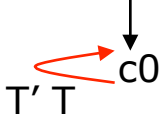
s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

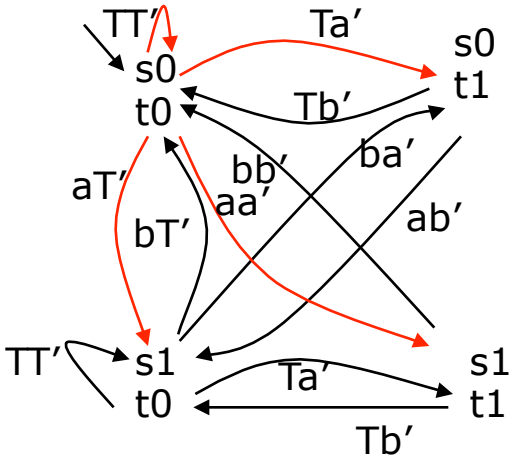
s0t0//c0′ ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s0t0//c0′ ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

Same formula state pair

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

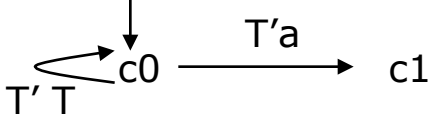
s0t0//c0' ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] } SUCCESS T in producer allowed

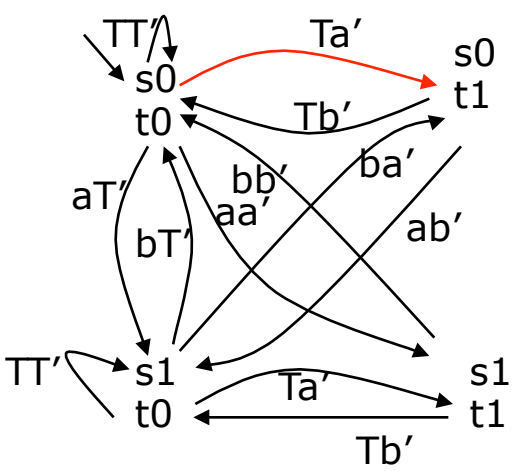
s0t1//c1 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s1t1//c2 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s1t0//c3 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

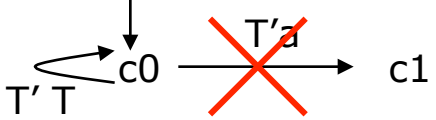
s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

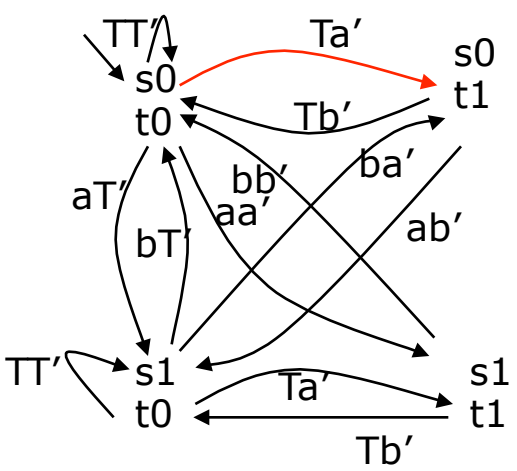
s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

s0t1//c1 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

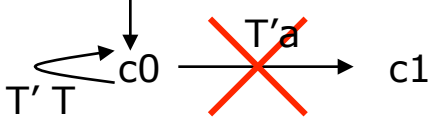
s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

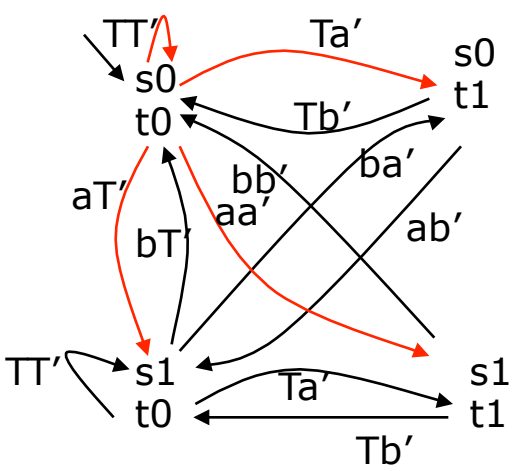
s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

s0t1//c1 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

FAIL

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

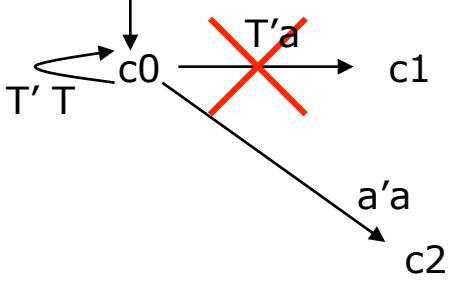
s0t0//c0' ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] } SUCCESS T in producer allowed

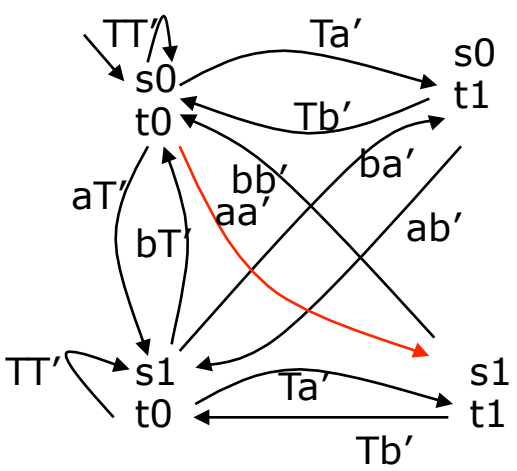
s0t1//c1 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] } FAIL T in producer blocked

s1t1//c2 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s1t0//c3 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

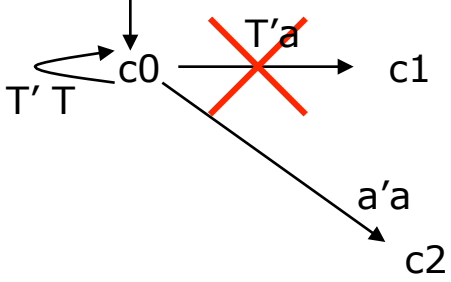
s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

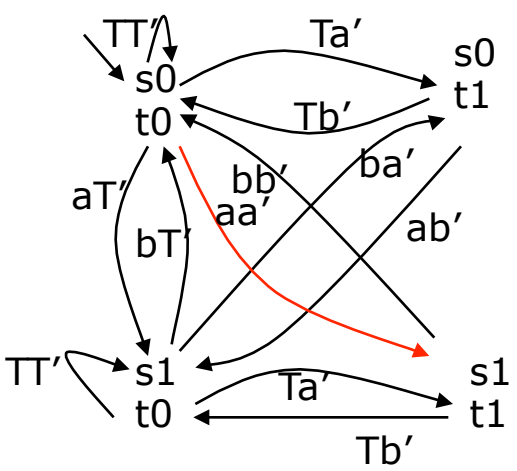
s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

s1t1//c2 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

Φ1 = s0t0⇒ AX¬(¬s1,t1)
Φ2 = s1t1⇒ AX¬(¬s0,t0)
Φ3 = s1t0⇒ AX¬(s0t0)





Example

s0t0//c0 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

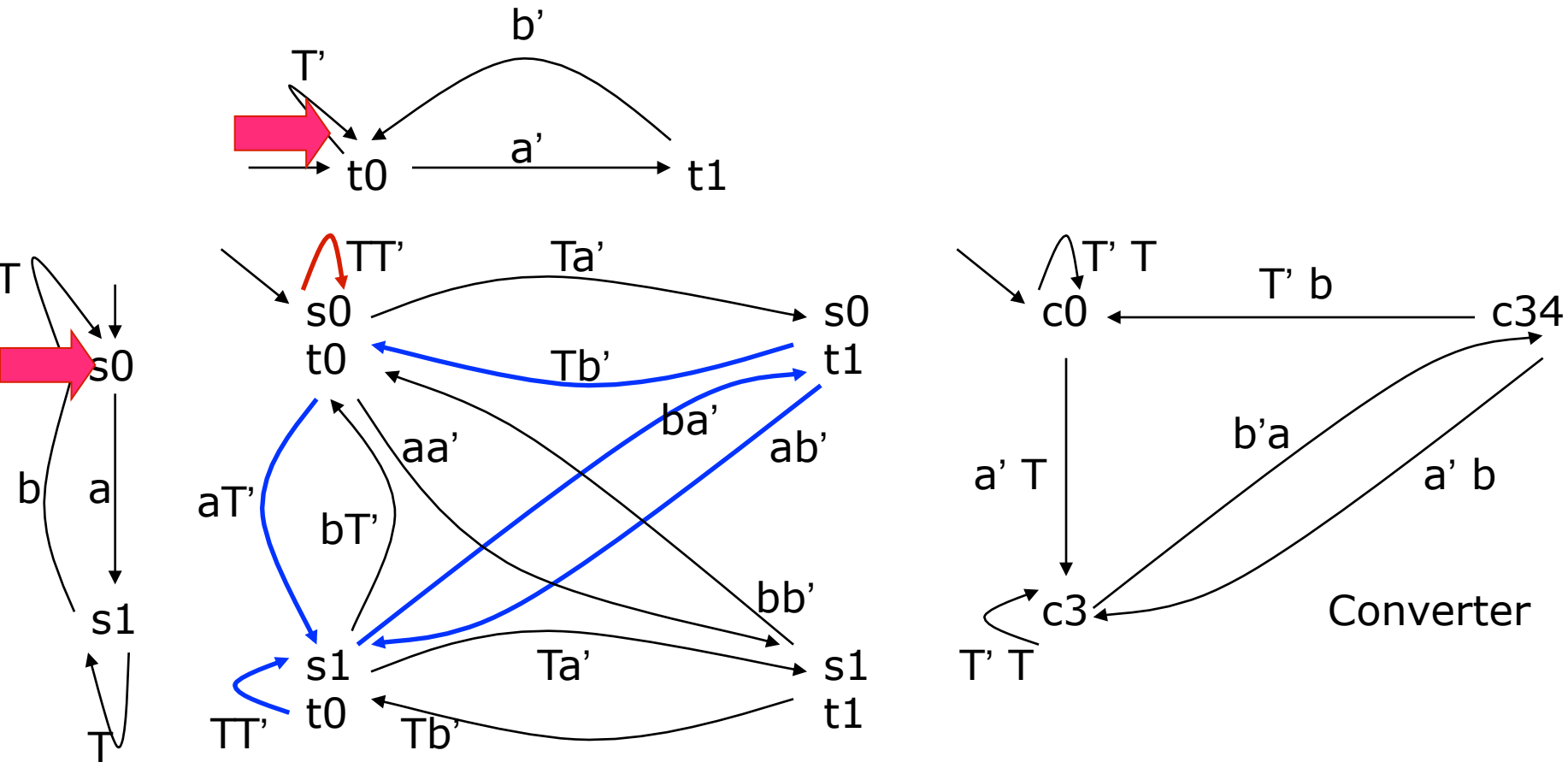
s0t0//c0 ⊨ {AXAG[Φ1], Φ1, AXAG[Φ2], Φ2, AXAG[Φ3], Φ3 }

s0t0//c0 ⊨ {AXAG[Φ1], AX¬(¬s1,t1), AXAG[Φ2], AXAG[Φ3] }

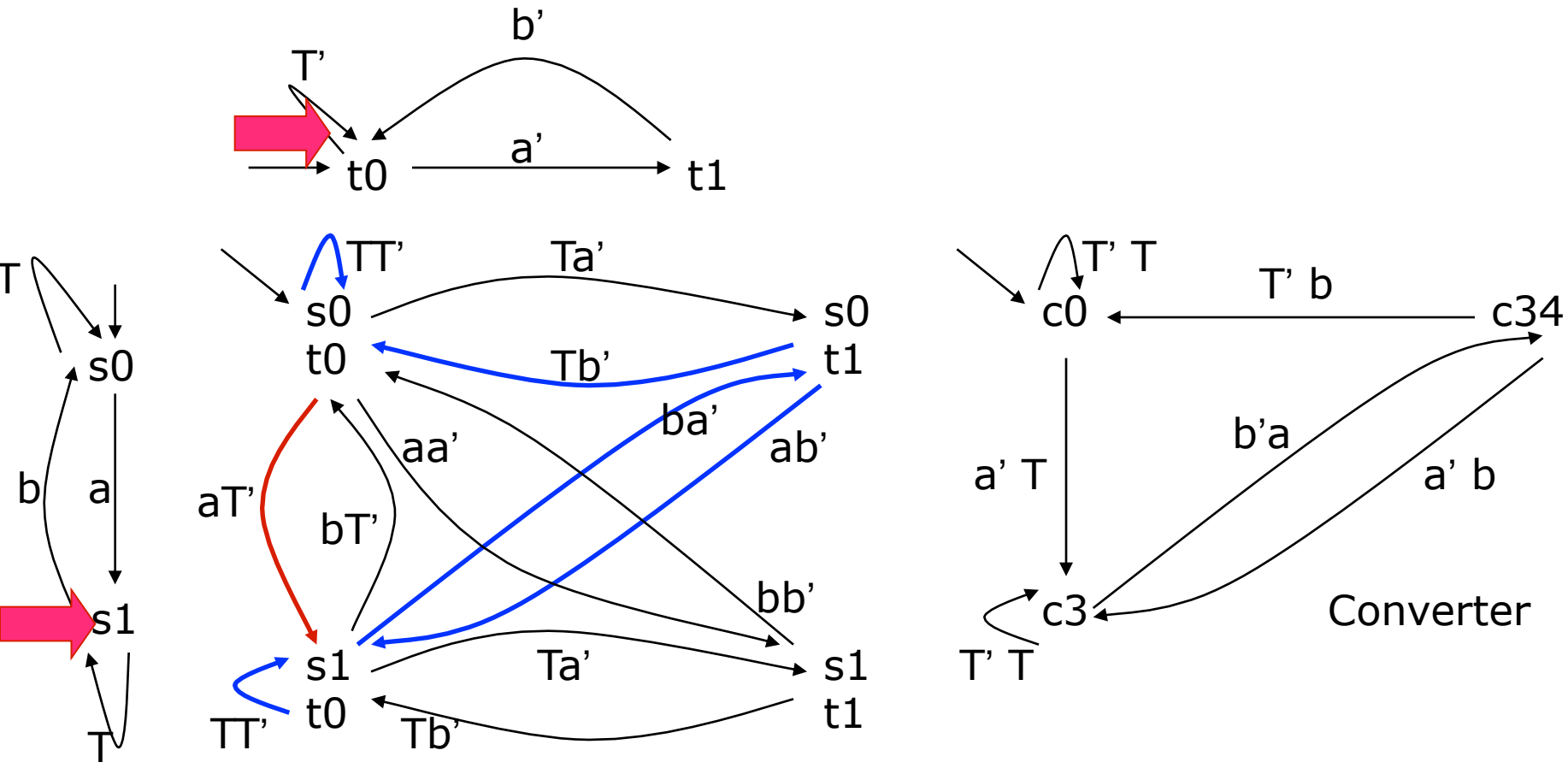
s1t1//c2 ⊨ {AG[Φ1], ¬(¬s1,t1), AG[Φ2], AG[Φ3] }

s1t1//c2 ⊨ {AG[Φ1], AG[Φ2], AG[Φ3] }

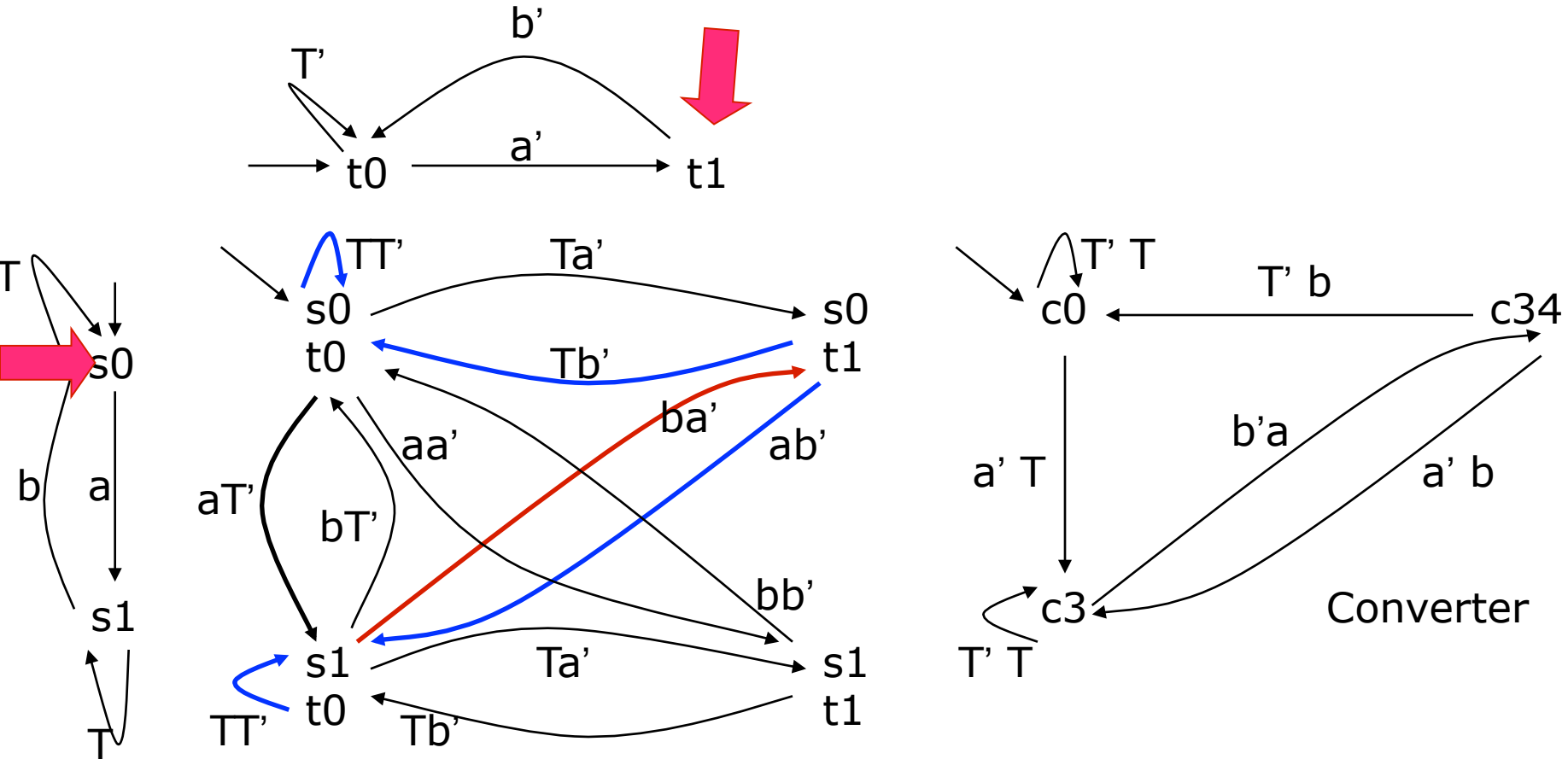
Example


b'
T'
a'
t0
t1
T
s0
b
a
s1
T
TT'
Ta'
s0
t0
s0
t1
Tb'
ba'
aa'
ab'
aT'
bT'
bb'
s1
t0
Ta'
s1
t1
TT'
Tb'
T' T
T' b
c0
c34
a' T
b'a
a' b
c3
T' T
Converter

Example


b'
T'
a'
t0
t1
T
s0
b
a
s1
T
TT'
Ta'
s0
t0
s0
t1
Tb'
ba'
aa'
ab'
aT'
bT'
bb'
s1
t0
Ta'
s1
t1
TT'
Tb'
T' T
T' b
c0
c34
a' T
b'a
a' b
c3
T' T
Converter

Example


b'
T'
a'
t0
t1
T
s0
b
a
s1
T
TT'
Ta'
s0
t0
s0
t1
Tb'
ba'
aa'
ab'
aT'
bT'
bb'
s1
t0
Ta'
s1
t1
TT'
Tb'
T' T
T' b
c0
c34
a' T
b'a
a' b
c3
Converter
T' T

Types of Protocol Mismatch

- Control mismatches
- Data Mismatches
- Clock mismatch

Conclusion

- Two key problems
  - Component selection / matching
  - Component composition
  - Both problems solved in the context of SoCs and SOC.
  - Key issues considered:
    - Control mismatches
    - Data-width / types
    - Clock
  - Future work: Incremental design

References

- Roop, Sowmya and Ramesh, “Forced Simulation – A Technique for Automating Component Reuse in Embedded Systems”, ACM-TODAES, October 2001.
- Robi Malik and Partha Roop, "Adaptive Techniques for Specification Matching in Embedded Systems: A Comparative Study", IFM 2005.
- Partha S. Roop, Arcot Sowmya, S. Ramesh, K-time forced Simulation: A Formal Verification Technique for IP Reuse. ICCD 2002: 50-55.
- Roopak Sinha, Partha S Roop and Samik Basu and Zoran Salcic, "A Module Checking based Converter Synthesis Approach for SOCs", VLSI Design 2008.
- Roopak Sinha, Partha S. Roop, Samik Basu, “SoC Design Approach Using Convertibility Verification”, EURASIP J. Emb. Sys. 2008.
- Roopak Sinha, Partha S Roop and Samik Basu and Zoran Salcic, “Multi-clock Soc design using protocol conversion”, DATE 2009.
- Roopak Sinha, Partha S Roop and Samik Basu and Zoran Salcic, ”Correct-by-construction multi-component SoC design” DATE 2012.
- Zachary J. Oster, Syed Adeel Ali, Ganesh Ram Santhanam, Samik Basu, Partha S. Roop: A Service Composition Framework Based on Goal-Oriented Requirements Engineering, Model Checking, and Qualitative Preference Analysis. ICSOC 2012: 283-297
- Syed Adeel Ali, Partha S. Roop, Ian Warren,: Web Service Choreography: Unanimous handling of Control and Data. International Journal of Software and Informatics (To Appear).

Additional slides

- The following slides discuss tableau construction to deal with data-width mismatches in SoCs followed by data-type mismatches in SOCs.

Complexity

$$O(|\mathbf{I}| \times 2^{|S|} \times 2^{|\Psi|} \times 2^{|\mathbf{E}|})$$

- |I| is the size of the set of all counter valuations:
  - For ***1*** counter $\mathbf{C}$ with range [0,R], there are R+3 valuations (R+1 valid values, 2 invalid)
  - For ***n*** counters where each counter $\mathbf{C}_i$'s range is $[0,R_i]$, $|I| = (R_1+3)x...x(R_{\boldsymbol{n}}+3)$.

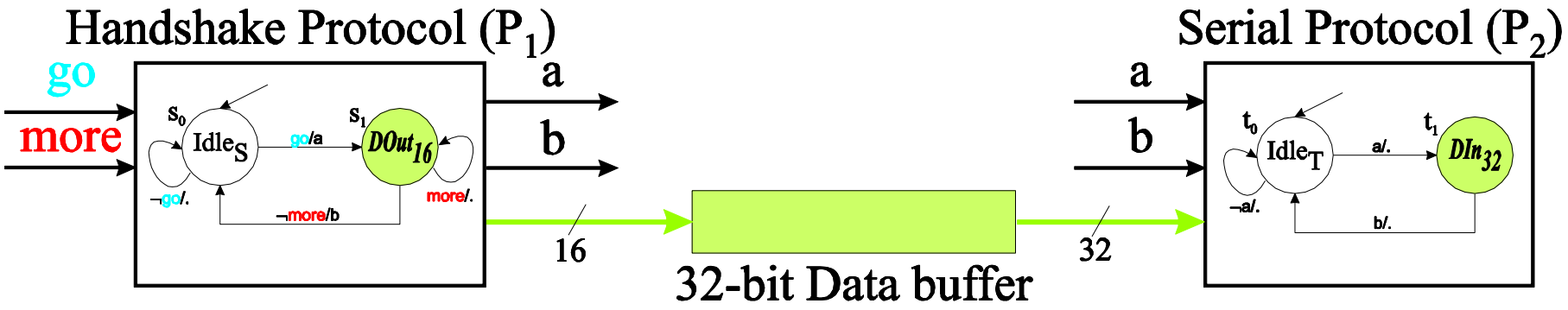
Complexity

$$O(|\mathrm{I}| \times 2^{|S|} \times 2^{|\Psi|} \times 2^{|\mathrm{E}|})$$

- |S| is the size of the synchronous parallel composition of all IPs.
- $|\Psi|$ is the size of the formula set Ψ.
- |E| is the size of the set of signals that can be buffered by the converter.

Introducing Data Counters

- P_1 and P_2 communicate using a 32-bit data buffer.
- P_1 writes 16-bit data ($DOut_{16}$) while P_2 reads 32-bit data (DIn_{32}).



Introducing Data Counters

- Data mismatches are possible:
  - P1 may write data when buffer is full (overflow).
  - P2 may read data when buffer is empty (underflow).
- Converter must ensure that the above situations are avoided.

Introducing Data Counters

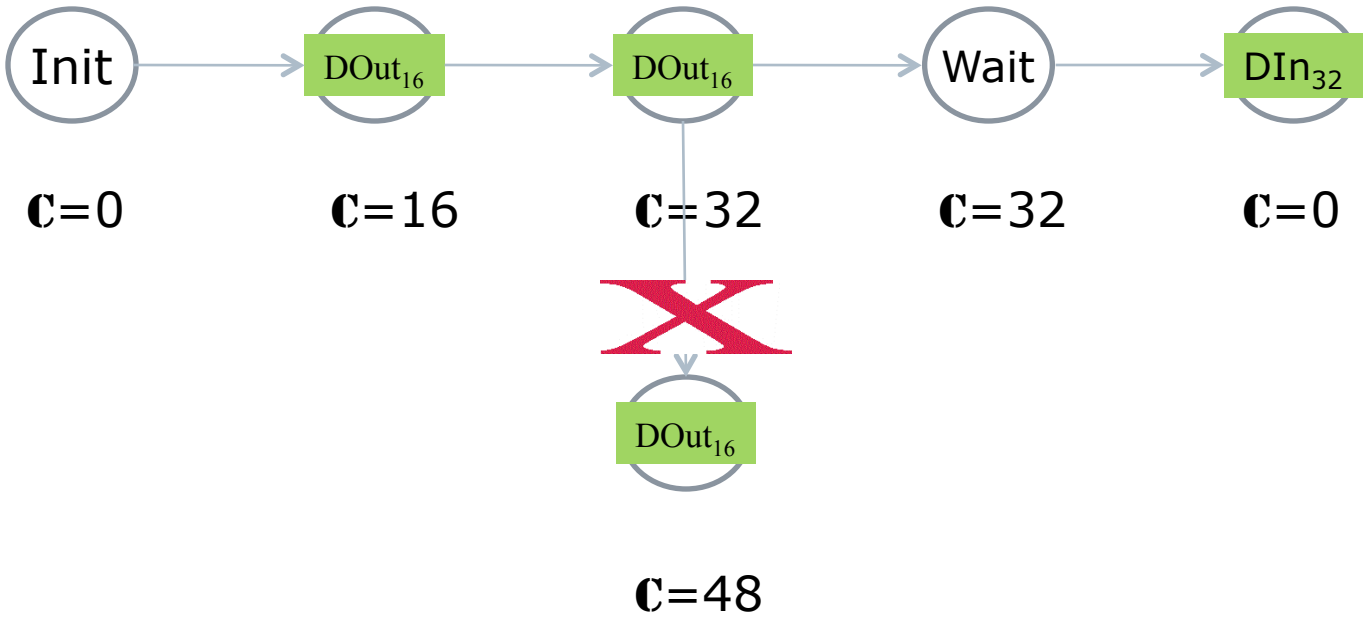
- We introduce a ***data counter*** $\mathfrak{C}$, which is used by the converter to keep track of the number of bits contained in the data buffer after *each transition* in the system. $\mathfrak{C}$ is initialized to 0.
- Whenever a $DOut_{16}$ is encountered, $\mathfrak{C}$ is incremented by 16.
- Whenever a DIn_{32} is encountered, $\mathfrak{C}$ is decremented by 32.

Introducing Data Counters

- The following CTL specification is used to ensure that counter remains within bounds

$$AG\,(0 \leq c \leq 32)$$

Processing Data Counters



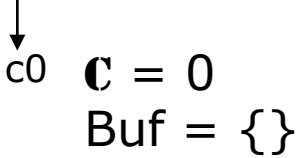
STEP-4: CTL Specifications

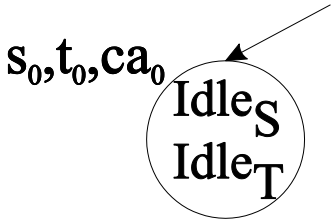
- AG EF $DOut_{16}$, AG EF DIn_{32} : There must always exist a reachable state in the system where P_1 can write data (P_2 can read data).
- AG AF ($Idle_S \wedge Idle_T \wedge \mathfrak{C}=0$): The protocols must always eventually reset to a state where the data buffer is empty.

STEP 5 – Model Checking

- Given the protocol composition and a set of properties, we can use tableau-construction as before to generate a converter.

$\Phi 1 = AG\ (0 \leq \mathbf{C} \leq 32)$
$\Phi 2 = AG\ EF\ DOut_{16}$
$\Phi 3 = AG\ EF\ DIn_{32}$
$\Phi 4 = AG\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)$

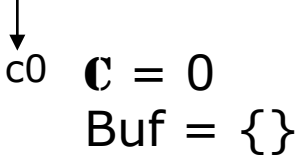


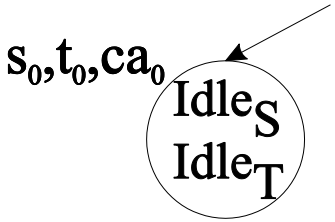


Example

C0//s0t0ca0 $\models$ {$\Phi 1$, $\Phi 2$, $\Phi 3$, $\Phi 4$}

$\Phi 1 = AG\ (0 \le \mathbf{C} \le 32)$
$\Phi 2 = AG\ EF\ DOut_{16}$
$\Phi 3 = AG\ EF\ DIn_{32}$
$\Phi 4 = AG\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)$





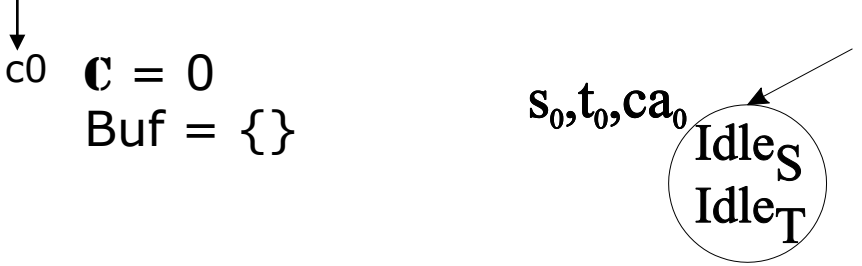
Example

$C0//s0t0ca0 \models \{\Phi 1\ ,\ \Phi 2\ ,\ \Phi 3,\ \Phi 4\}$

UNR tableau rule

$C0//s0t0ca0 \models \{(0 \le \mathbf{C} \le 32),\ AX\ \Phi 1\ ,\ EF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ EF\ DIn_{32},\ AX\ \Phi 4,$
$AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

$\Phi 1 = AG\ (0 \le \mathbf{C} \le 32)$
$\Phi 2 = AG\ EF\ DOut_{16}$
$\Phi 3 = AG\ EF\ DIn_{32}$
$\Phi 4 = AG\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)$



Example

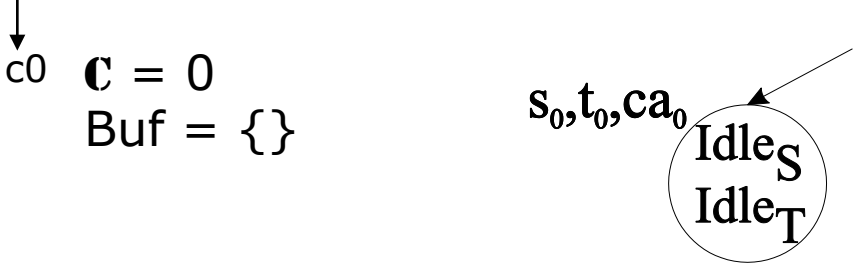
$C0//s0t0ca0 \models \{\Phi 1\ ,\ \Phi 2\ ,\ \Phi 3,\ \Phi 4\}$

$C0//s0t0ca0 \models \{(0 \le \mathbf{C} \le 32),\ AX\ \Phi 1\ ,\ EF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ EF\ DIn_{32},\ AX\ \Phi 4,$
$AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

UNR tableau rule

$C0//s0t0ca0 \models \{AX\ \Phi 1\ ,\ DOut_{16} \vee EXEF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ DIn_{32} \vee EXEF\ DIn_{32},\ AX\ \Phi 4,$
$(Idle_S \wedge Idle_T \wedge \mathbf{C}=0) \vee AX\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

$\Phi 1 = AG\ (0 \le \mathbf{C} \le 32)$
$\Phi 2 = AG\ EF\ DOut_{16}$
$\Phi 3 = AG\ EF\ DIn_{32}$
$\Phi 4 = AG\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)$



Example

$C0//s0t0ca0 \models \{\Phi 1\ ,\ \Phi 2\ ,\ \Phi 3,\ \Phi 4\}$

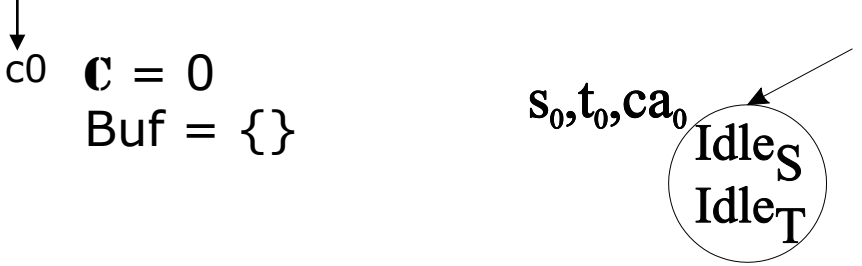
$C0//s0t0ca0 \models \{(0 \le \mathbf{C} \le 32),\ AX\ \Phi 1\ ,\ EF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ EF\ DIn_{32},\ AX\ \Phi 4,$
$AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

$C0//s0t0ca0 \models \{AX\ \Phi 1\ ,\ DOut_{16} \vee EXEF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ DIn_{32} \vee EX\ EF\ DIn_{32},\ AX\ \Phi 4,$
$(Idle_S \wedge Idle_T \wedge \mathbf{C}=0) \vee AX\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

OR tableau rule

$C0//s0t0ca0 \models \{AX\ \Phi 1\ ,\ EXEF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ EX\ EF\ DIn_{32},\ AX\ \Phi 4,$
$(Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

$\Phi 1 = AG\ (0 \le \mathbf{C} \le 32)$
$\Phi 2 = AG\ EF\ DOut_{16}$
$\Phi 3 = AG\ EF\ DIn_{32}$
$\Phi 4 = AG\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)$



Example

$C0//s0t0ca0 \models \{\Phi 1\ ,\ \Phi 2\ ,\ \Phi 3,\ \Phi 4\}$

$C0//s0t0ca0 \models \{(0 \le \mathbf{C} \le 32),\ AX\ \Phi 1\ ,\ EF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ EF\ DIn_{32},\ AX\ \Phi 4,$
$AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

$C0//s0t0ca0 \models \{AX\ \Phi 1\ ,\ DOut_{16} \vee EXEF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ DIn_{32} \vee EX\ EF\ DIn_{32},\ AX\ \Phi 4,$
$(Idle_S \wedge Idle_T \wedge \mathbf{C}=0) \vee AX\ AF\ (Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

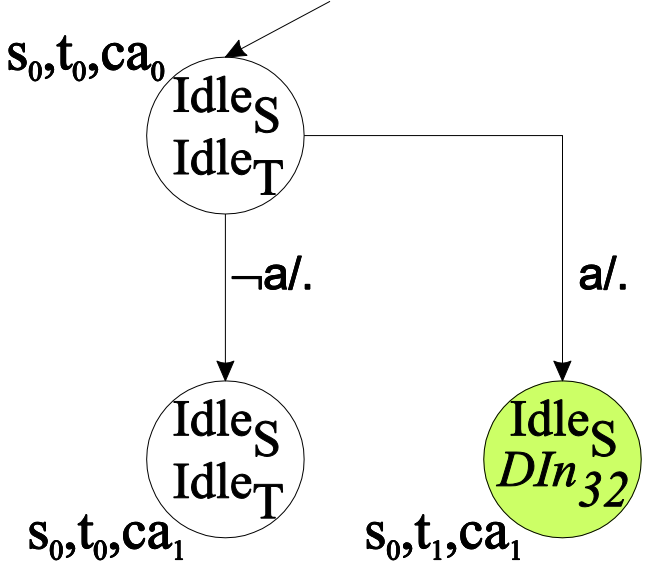
$C0//s0t0ca0 \models \{AX\ \Phi 1\ ,\ EXEF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ EX\ EF\ DIn_{32},\ AX\ \Phi 4,$
$(Idle_S \wedge Idle_T \wedge \mathbf{C}=0)\}$

$C0//s0t0ca0 \models \{AX\ \Phi 1\ ,\ EXEF\ DOut_{16}\ ,\ AX\ \Phi 2\ ,\ AX\ \Phi 3,\ EX\ EF\ DIn_{32},\ AX\ \Phi 4\}$

$\Phi 1$ = AG ($0 \leq \mathbf{c} \leq 32$)
$\Phi 2$ = AG EF $DOut_{16}$
$\Phi 3$ = AG EF DIn_{32}
$\Phi 4$ = AG AF ($Idle_S \wedge Idle_T \wedge \mathbf{c}=0$)

Example

c0 $\mathbf{c} = 0$
Buf = {}

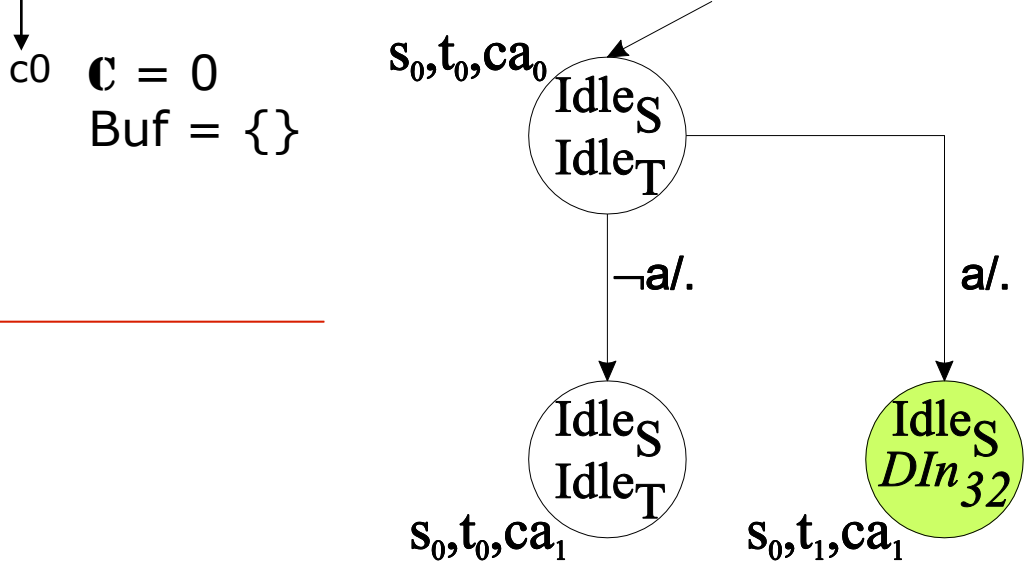


C0//s0t0ca0 ⊨ {AX $\Phi 1$, EXEF $DOut_{16}$, AX $\Phi 2$, AX $\Phi 3$, EX EF DIn_{32}, AX $\Phi 4$}

Ψ_{AX} = {$\Phi 1$, $\Phi 2$, $\Phi 3$, $\Phi 4$}
Ψ_{EX} = {EF $DOut_{16}$, EF DIn_{32}}

$\Phi 1 = AG\ (0 \le \mathfrak{C} \le 32)$
$\Phi 2 = AG\ EF\ DOut_{16}$
$\Phi 3 = AG\ EF\ DIn_{32}$
$\Phi 4 = AG\ AF\ (Idle_S \wedge Idle_T \wedge \mathfrak{C}=0)$

Example



C0//s0t0ca0 ⊨ {AX Φ1 , EXEF $DOut_{16}$, AX Φ2 , AX Φ3, EX EF DIn_{32}, AX Φ4}

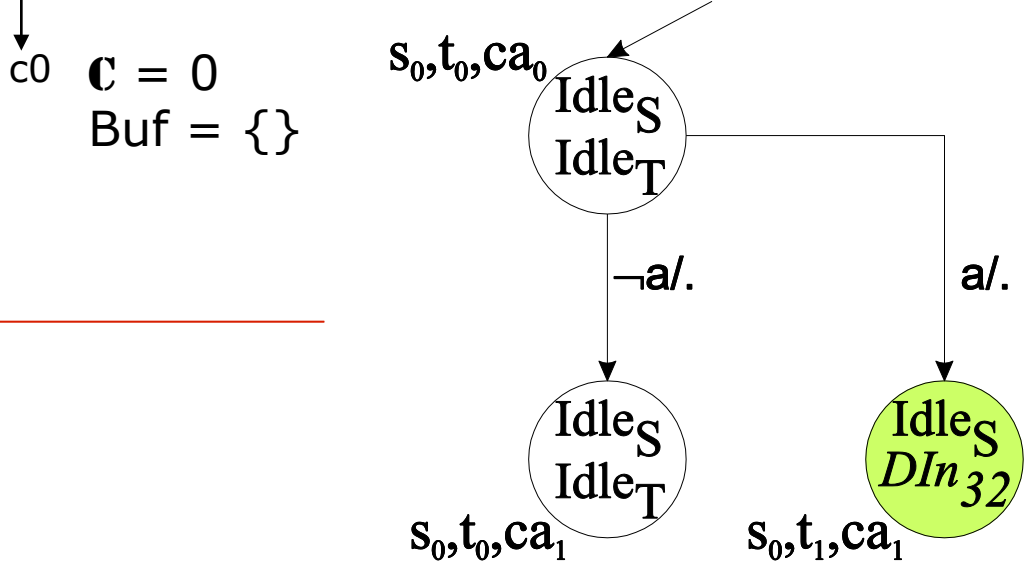
$\Psi_{AX} = \{\Phi 1, \Phi 2, \Phi 3, \Phi 4\}$
$\Psi_{EX} = \{EF\ DOut_{16}, EF\ DIn_{32}\}$

$\Pi = \{(s_0,t_0,ca_1), (s_0,t_1,ca_1)\}$

$\pi \subseteq \Pi = \{(s_0,t_0,ca_1), (s_0,t_1,ca_1)\}$

$\Phi 1 = AG\ (0 \le \mathfrak{C} \le 32)$
$\Phi 2 = AG\ EF\ DOut_{16}$
$\Phi 3 = AG\ EF\ DIn_{32}$
$\Phi 4 = AG\ AF\ (Idle_S \wedge Idle_T \wedge \mathfrak{C}=0)$

Example



C0//s0t0ca0 ⊨ {AX Φ1 , EXEF $DOut_{16}$, AX Φ2 , AX Φ3, EX EF DIn_{32}, AX Φ4}

$\Psi_{AX} = \{\Phi 1, \Phi 2, \Phi 3, \Phi 4\}$
$\Psi_{EX} = \{EF\ DOut_{16}, EF\ DIn_{32}\}$

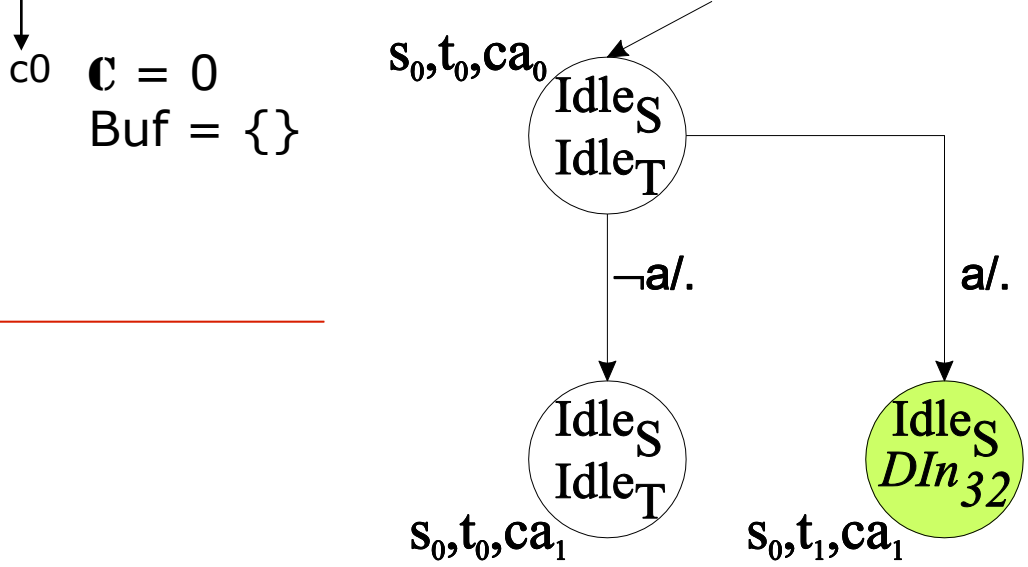
$\Pi = \{(s_0,t_0,ca_1), (s_0,t_1,ca_1)\}$

$\pi \subseteq \Pi = \{(s_0,t_0,ca_1), (s_0,$ 

- Signal a is not present in buffers.
- Transition to (s_0,t_1,ca_1) will lead to counter value to become negative.

$\Phi 1$ = AG ($0 \leq \mathbf{c} \leq 32$)
$\Phi 2$ = AG EF $DOut_{16}$
$\Phi 3$ = AG EF DIn_{32}
$\Phi 4$ = AG AF ($Idle_S \wedge Idle_T \wedge \mathbf{c}=0$)

Example



C0//s0t0ca0 ⊨ {AX $\Phi 1$, EXEF $DOut_{16}$, AX $\Phi 2$, AX $\Phi 3$, EX EF DIn_{32}, AX $\Phi 4$}

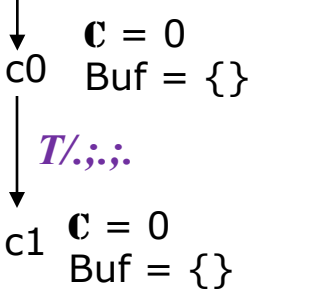
Ψ_{AX} = {$\Phi 1$, $\Phi 2$, $\Phi 3$, $\Phi 4$}
Ψ_{EX} = {EF $DOut_{16}$, EF DIn_{32}}

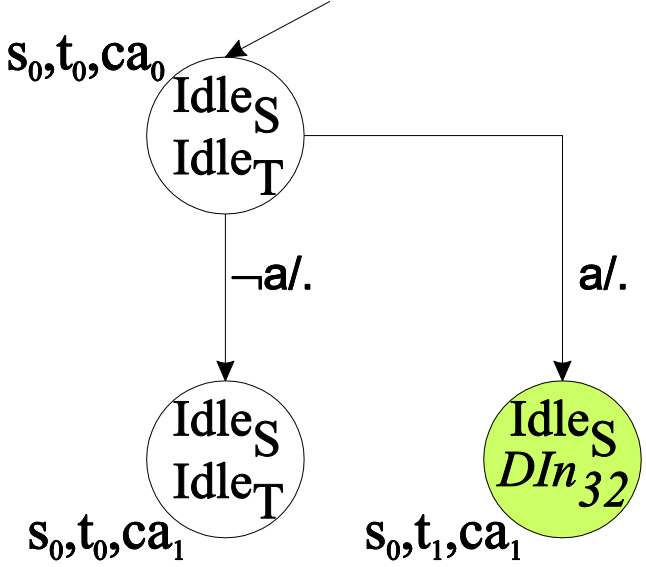
Π= {(s_0,t_0,ca_1), (s_0,t_1,ca_1)}

$\pi \subseteq \Pi$ = {(s_0,t_0,ca_1)}

$\Phi 1$ = AG ($0 \le \mathbf{c} \le 32$)
$\Phi 2$ = AG EF $DOut_{16}$
$\Phi 3$ = AG EF DIn_{32}
$\Phi 4$ = AG AF ($Idle_S \wedge Idle_T \wedge \mathbf{c}=0$)

Example





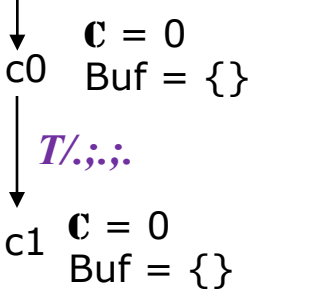
C0//s0t0ca0 ⊨ {AX $\Phi 1$, EXEF $DOut_{16}$, AX $\Phi 2$, AX $\Phi 3$, EX EF DIn_{32}, AX $\Phi 4$}

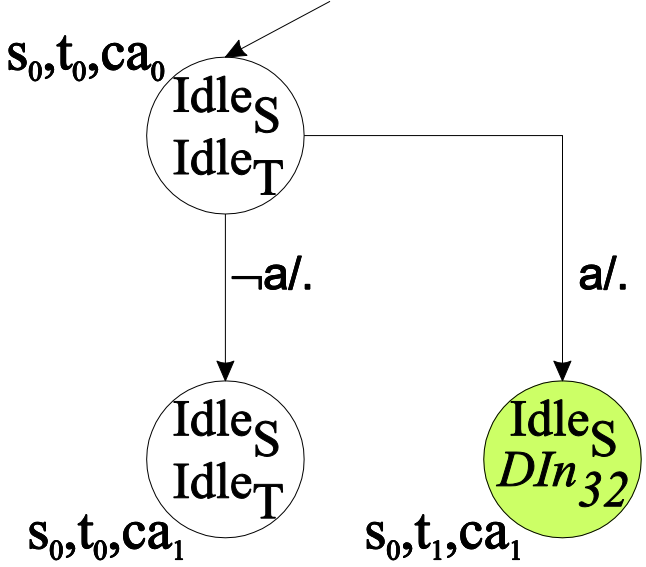
$\Psi_{AX} = \{\Phi 1, \Phi 2, \Phi 3, \Phi 4\}$
$\Psi_{EX} = \{EF\ DOut_{16}, EF\ DIn_{32}\}$
$\Pi = \{(s_0,t_0,ca_1), (s_0,t_1,ca_1)\}$
$\pi \subseteq \Pi = \{(s_0,t_0,ca_1)\}$

C1//s0t0ca1 ⊨ {$\Phi 1$, EF $DOut_{16}$, $\Phi 2$, $\Phi 3$, EF DIn_{32}, $\Phi 4$}

$\Phi 1$ = AG ($0 \le \mathbf{c} \le 32$)
$\Phi 2$ = AG EF $DOut_{16}$
$\Phi 3$ = AG EF DIn_{32}
$\Phi 4$ = AG AF ($Idle_S \wedge Idle_T \wedge \mathbf{c}=0$)

Example



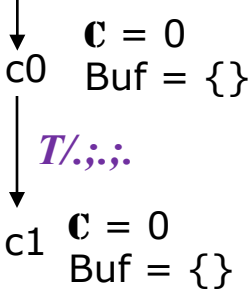


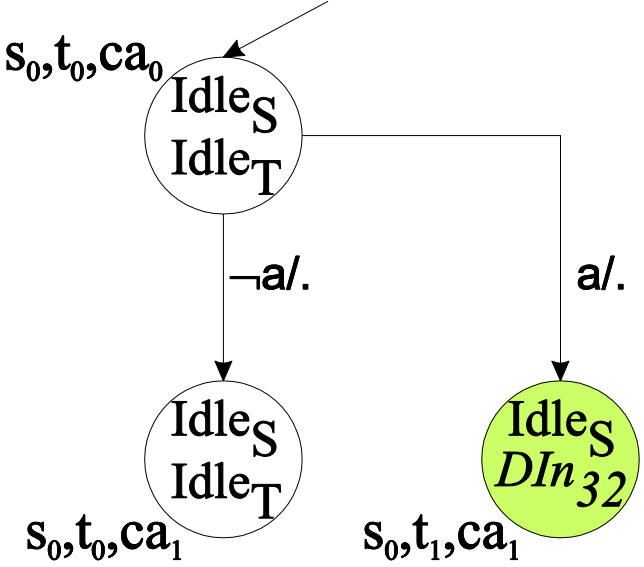
C1//s0t0ca1 ⊨ {$\Phi 1$, EF $DOut_{16}$, $\Phi 2$, $\Phi 3$, EF DIn_{32}, $\Phi 4$}

C1//s0t0ca1 ⊨ {($0 \le \mathbf{c} \le 32$) , AX$\Phi 1$, $DOut_{16}$ ∨ EX EF $DOut_{16}$, AX $\Phi 2$, EF DIn_{32} , AX $\Phi 3$,
AF ($Idle_S \wedge Idle_T \wedge \mathbf{c}=0$), AX $\Phi 4$}

$\Phi 1$ = AG ($0 \le \mathbf{c} \le 32$)
$\Phi 2$ = AG EF $DOut_{16}$
$\Phi 3$ = AG EF DIn_{32}
$\Phi 4$ = AG AF ($Idle_S \wedge Idle_T \wedge \mathbf{c}=0$)

Example





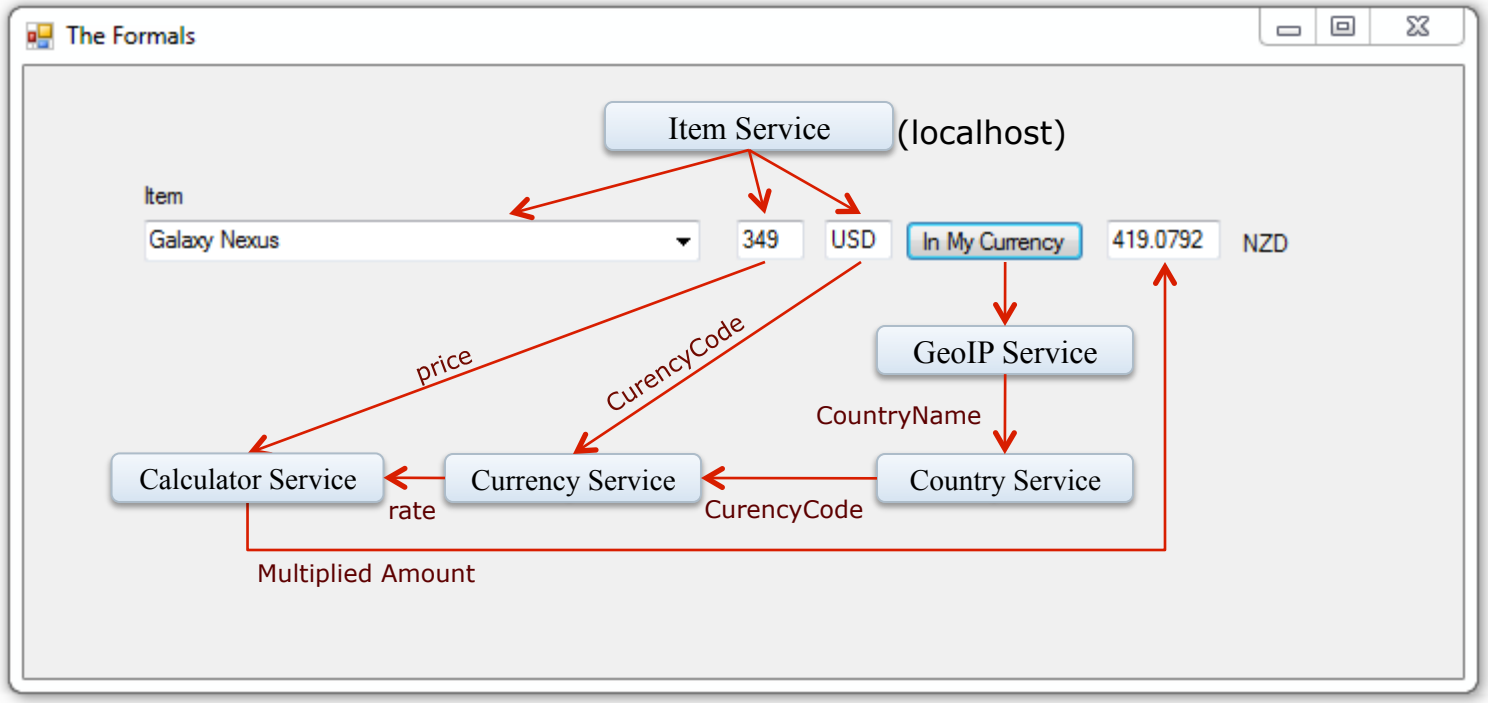
C1//s0t0ca1 $\models$ {$\Phi 1$, EF $DOut_{16}$, $\Phi 2$, $\Phi 3$, EF DIn_{32}, $\Phi 4$}

C1//s0t0ca1 $\models$ {($0 \le \mathbf{c} \le 32$) , AX$\Phi 1$, $DOut_{16} \vee$ EX EF $DOut_{16}$, AX $\Phi 2$, EF DIn_{32} , AX $\Phi 3$,
AF ($Idle_S \wedge Idle_T \wedge \mathbf{c}=0$), AX $\Phi 4$}

and so on....

A Too for SoC Composition

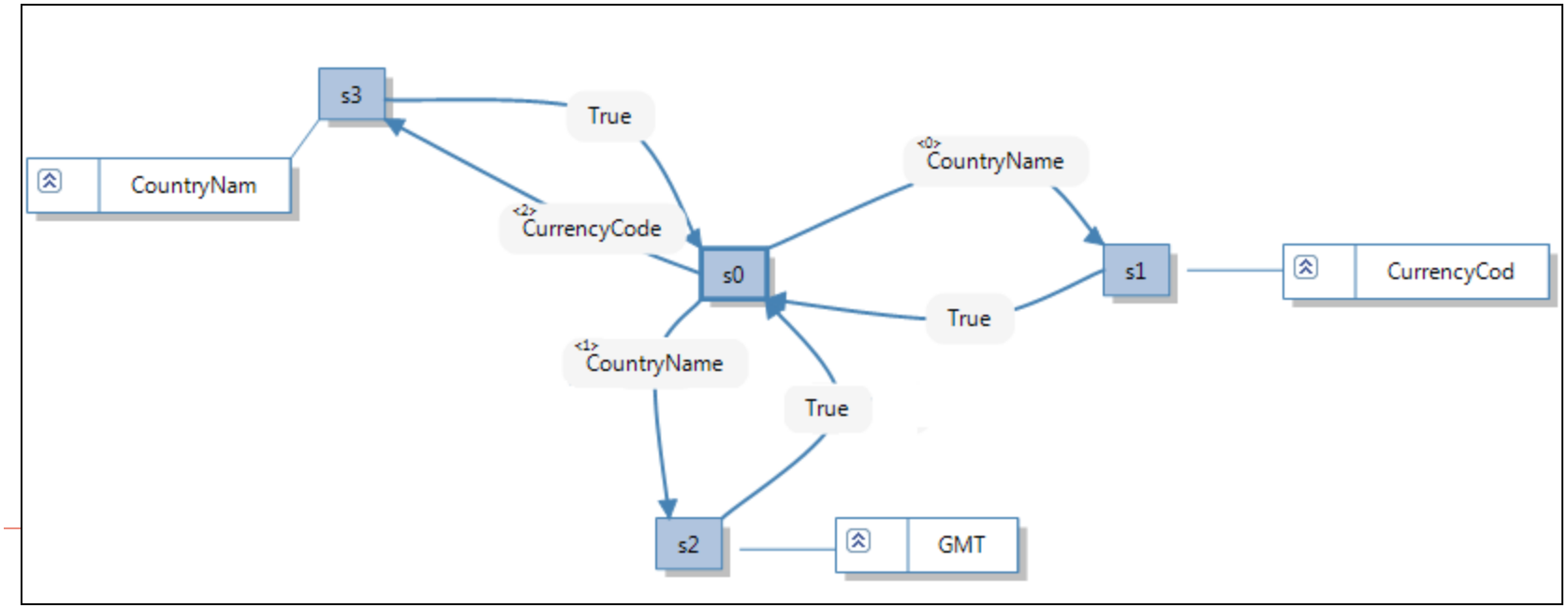
The currency converter revisited



Click for Demo

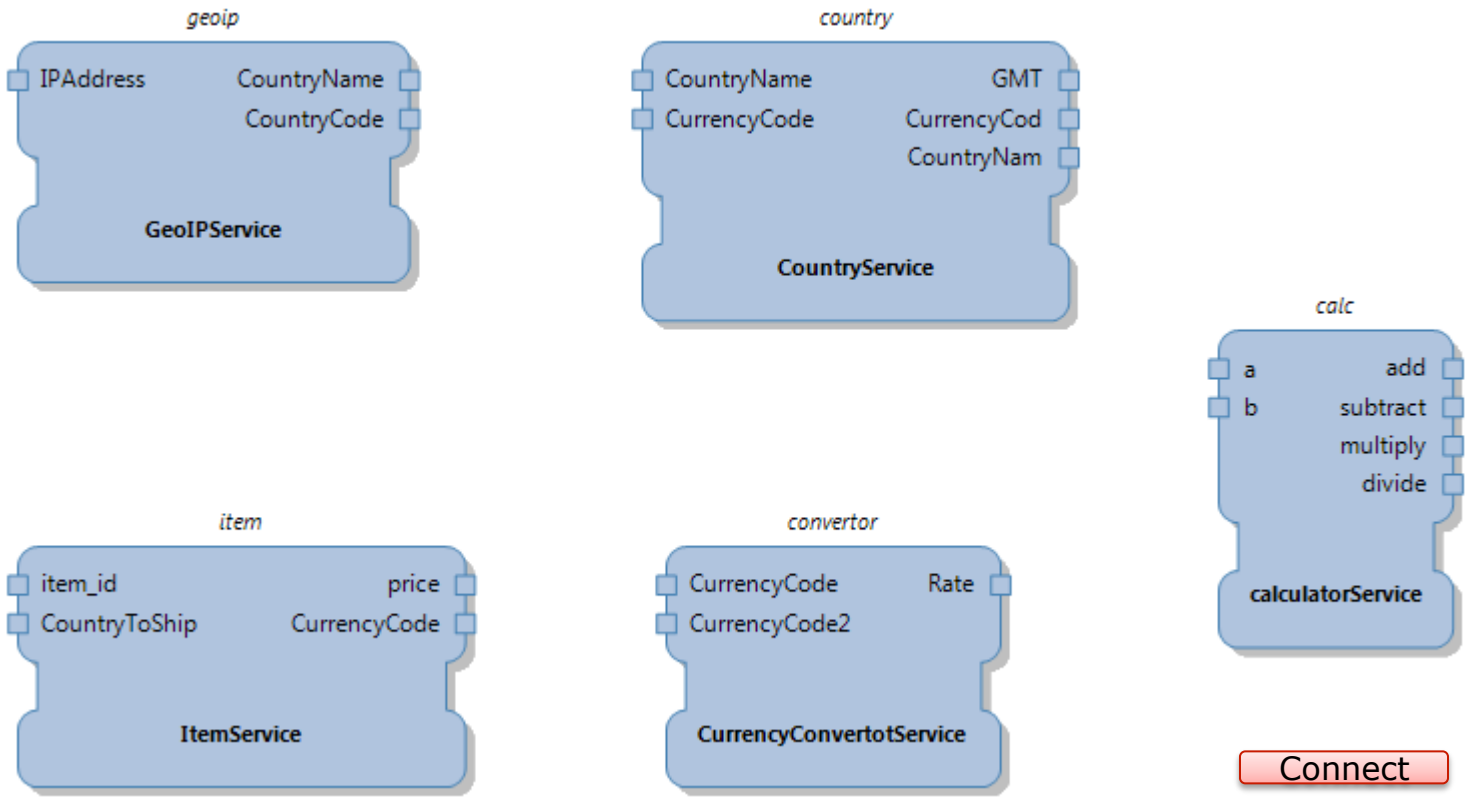
Auto-FSM via WSDL

Country Service - http://www.webservicex.net/country.asmx?WSDL

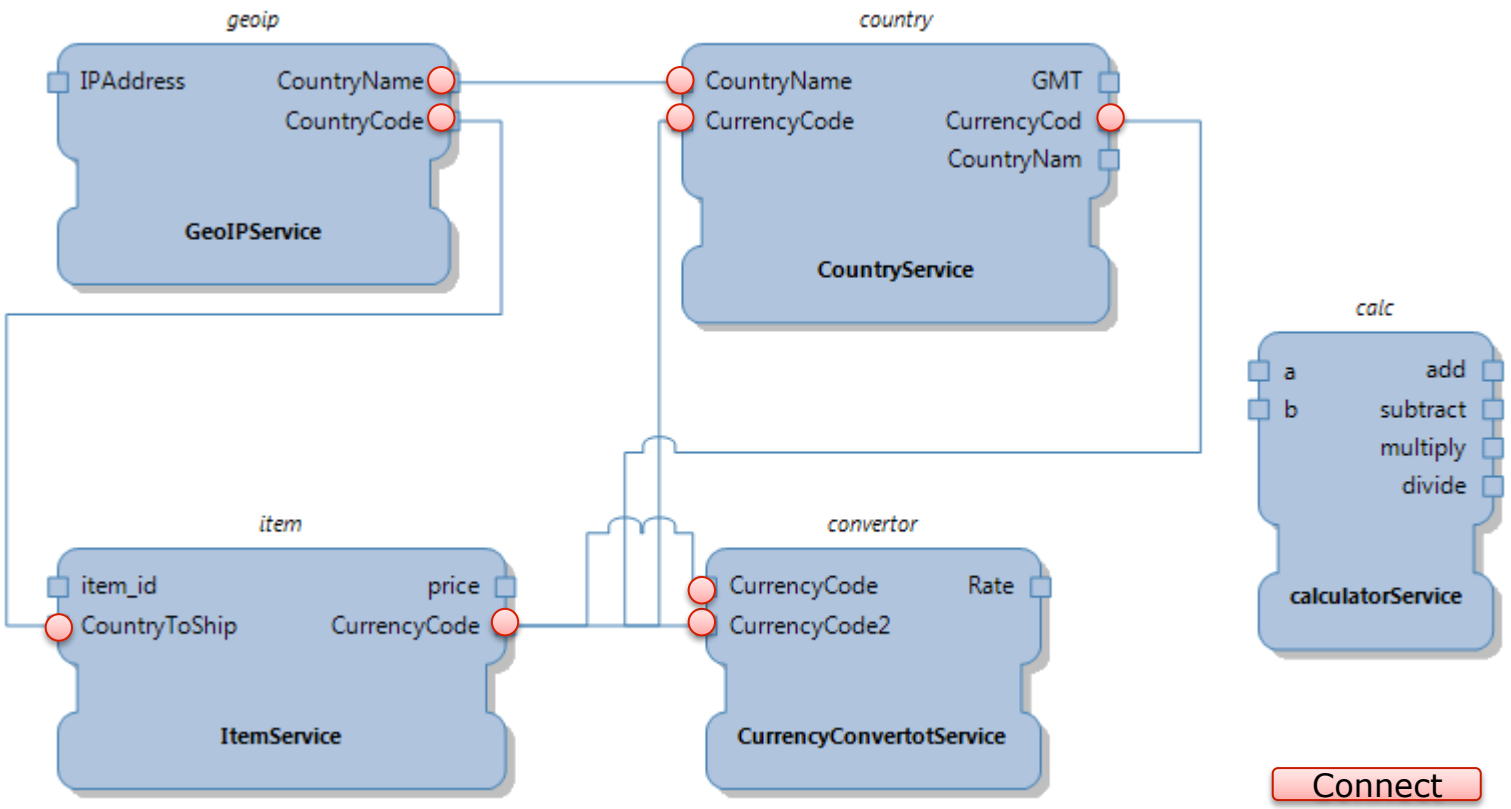


Currency Conv Example

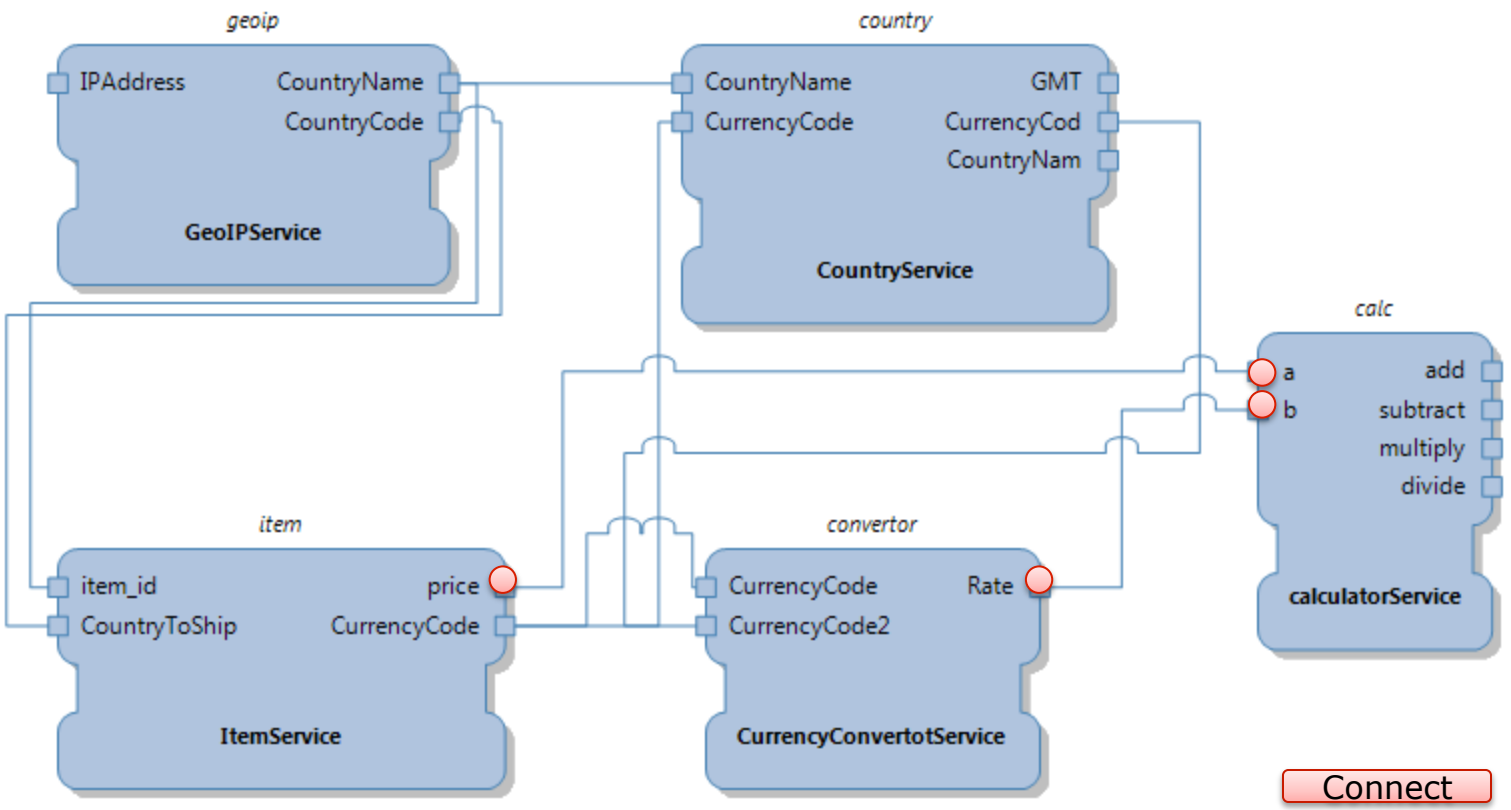
GeoIP Service - http://www.webservicex.net/geoipservice.asmx?WSDL
Calculator Service www.html2xml.nl/Services/Calculator/Version1/Calculator.asmx?WSDL
Country Service - http://www.webservicex.net/country.asmx?WSDL
Currency Service - http://www.webservicex.net/CurrencyConvertor.asmx?WSDL
Item Service – localhost:80



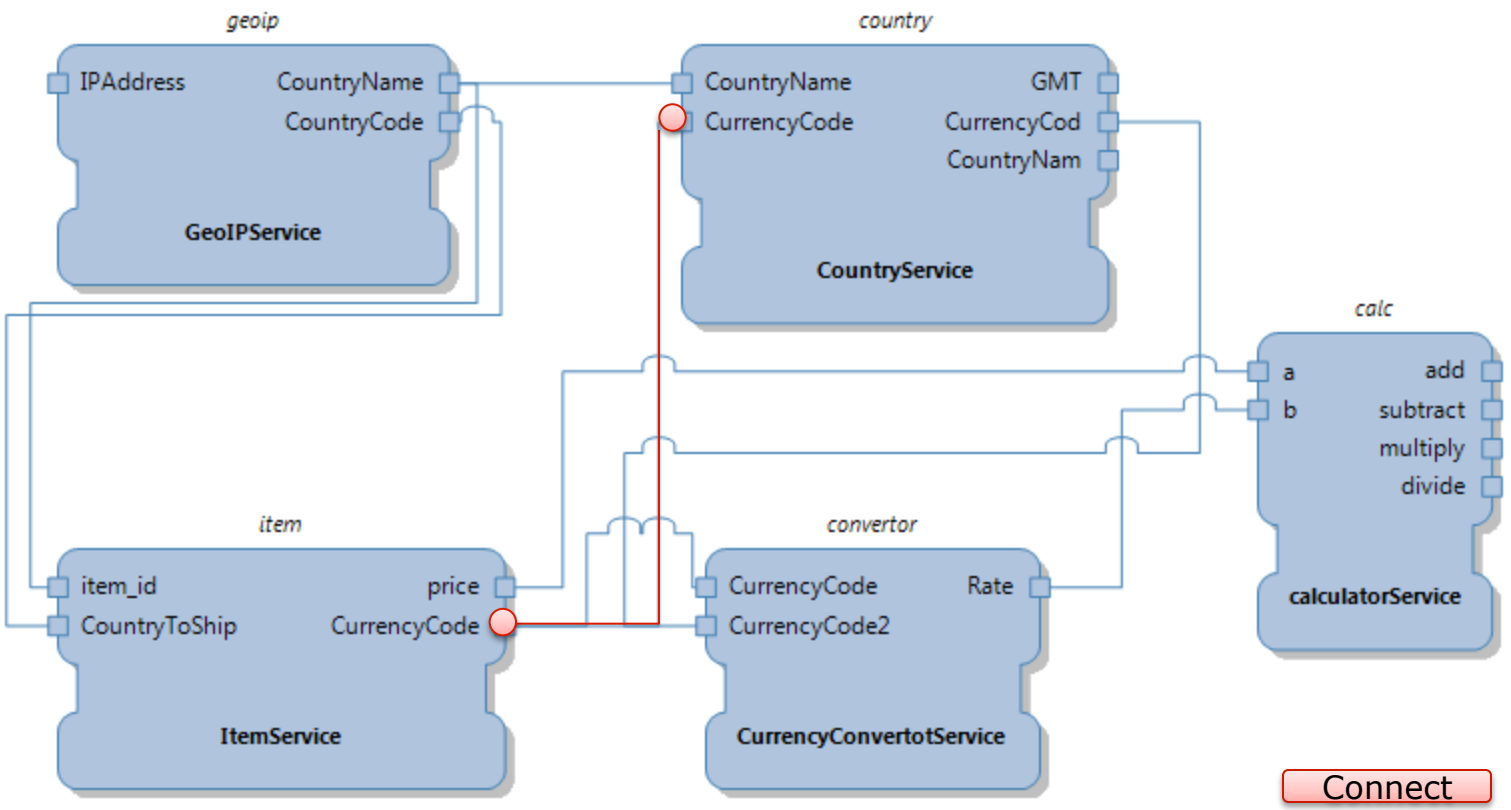
Auto Connect



Auto+Manual Connect



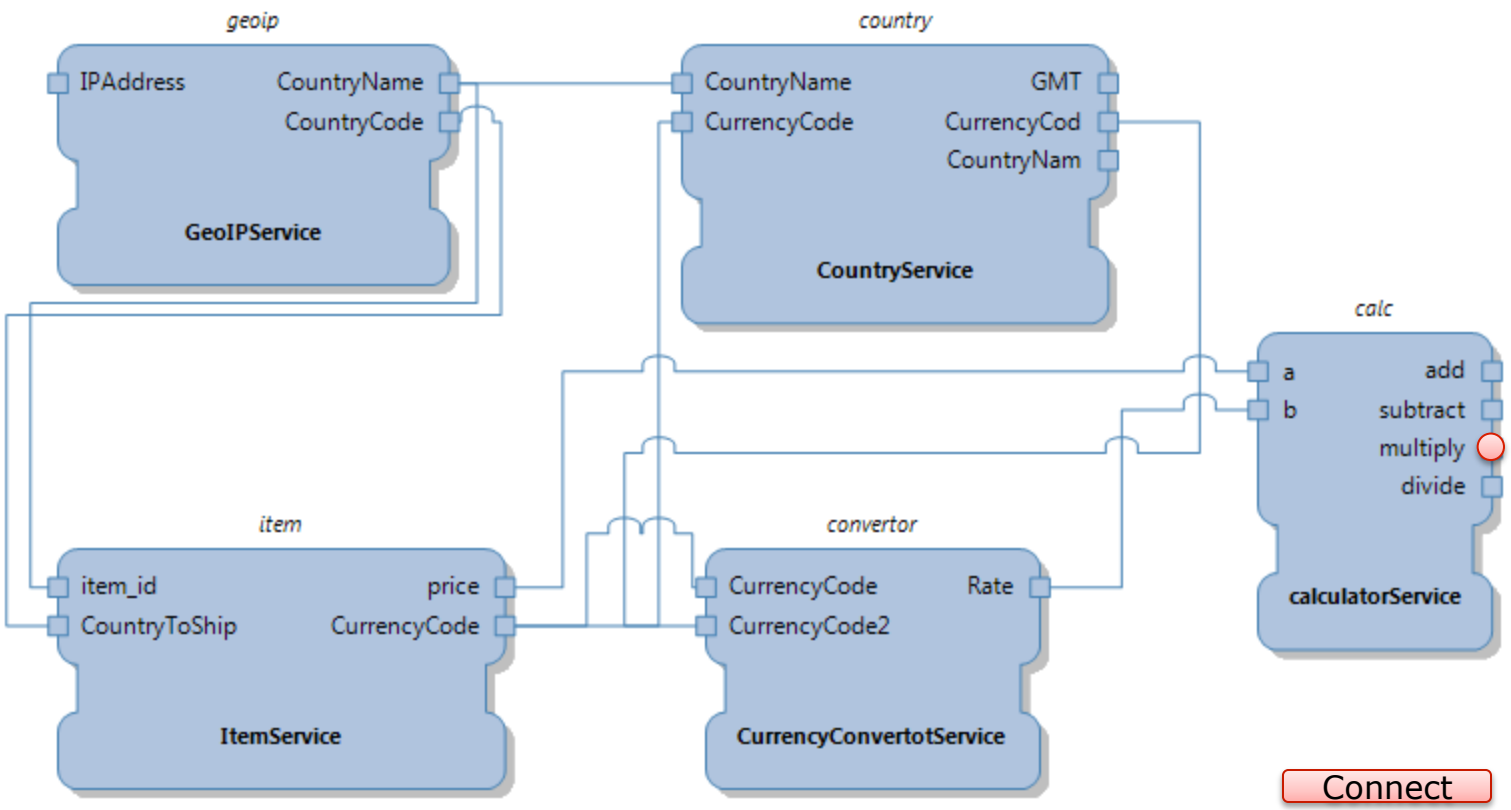
Redundant Connections



Goal specifications

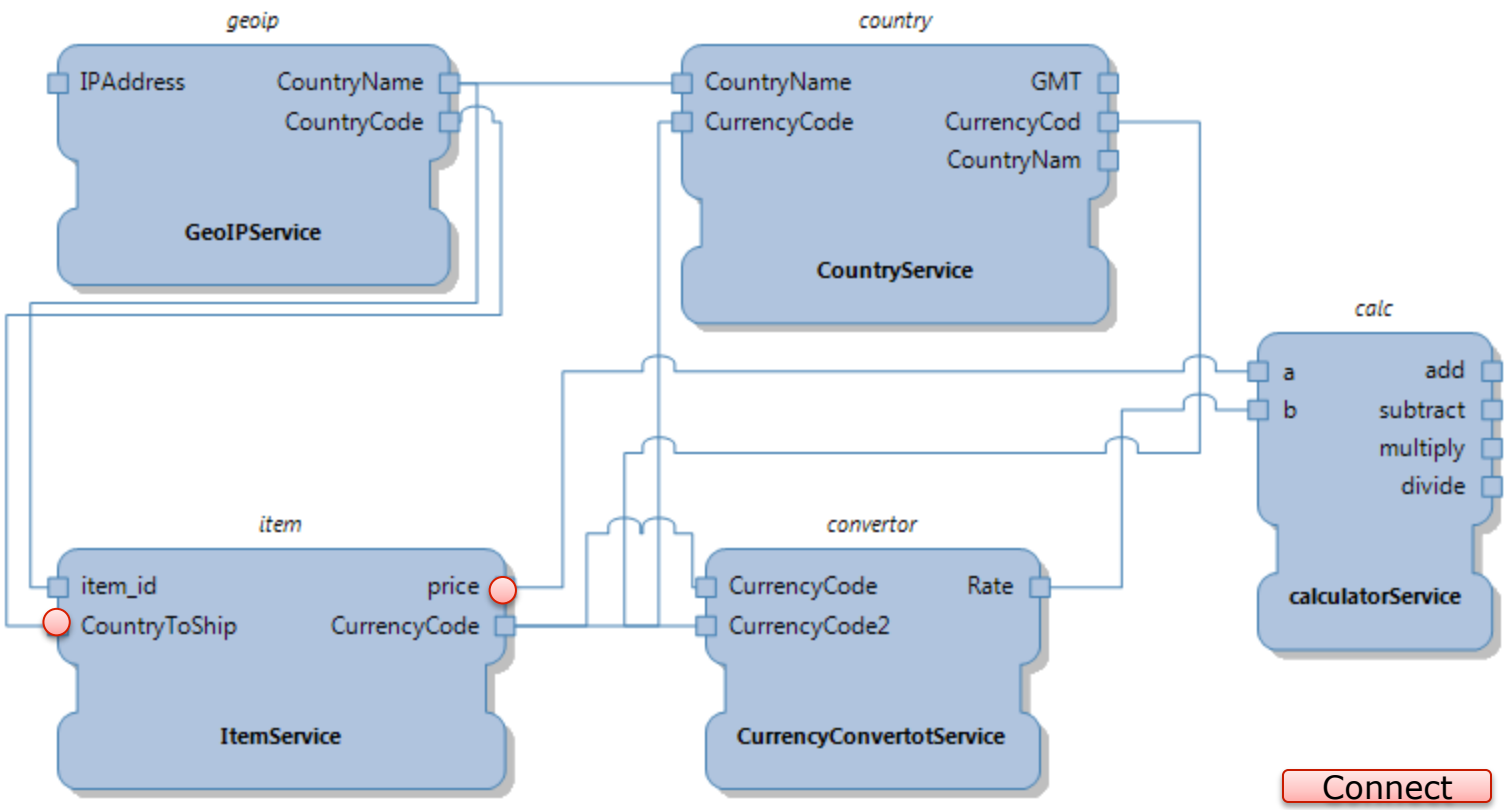
- The price must not be calculated until destination country is known.
- Conversion should me made from **item's** currency to **user's** currency.
- There must exist a path to a state where the convertered rate can be obtained.

Specifying the Goal



GOAL: Obtain the converted rate
CTL: EF(Label=calc.multiply)

Specifying the Goal



Constraint 1: The price must not be calculated until destination country is known.

CTL: ~(Label=item.price)AU(Label=item.CountryToShip)