

Synthesizing Glue Operators from Glue Constraints for the Construction of Component-Based Systems

Simon Bludze and Joseph Sifakis

cea **list**



Zürich, June 30th, 2011

Outline

- Motivation
- BIP and the Glue
- Synthesizing glue operators
- Design flow

Quite some liberties taken w.r.t. the paper for the sake of the presentation clarity!

Outline

- Motivation
- BIP and the Glue
- Synthesizing glue operators
- Design flow



At the TOOLS keynote on Tuesday...

...**Oscar Nierstrasz** spoke of the necessity of

- Manipulating the models
- Bridging the gap between high-level models and run-time code

Questions:

- Recently, did we get any closer to these objectives? If not, what is the way there?
- Does not raising the abstraction level rather increase the gap?

Answer:

- We should build *solid* and *light-weight* bridges!



Programming is Modeling



Bridging the gaps



Solid and light-weight bridges

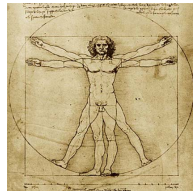
A unified modelling formalism

Solid:

- Clearly established formal semantics
- Heterogeneity
 - computation, execution, implementation
- Certifying code generation

Light-weight:

- Clear, accessible formal semantics
- Minimal set of primitives
- Separation of concerns
 - coordination is a first-class citizen
- Efficient implementation for popular platforms



More specifically

Context: *Component-based* modelling, design and validation of embedded (safety-critical) systems.

Presently:

- A number of *coordination mechanisms* for concurrent systems
 - shared variables, semaphores, message passing, etc.
- Ad-hoc use and analysis methodologies.

Our goal: *Unified framework* for component-based modelling and design

- Incremental description
- Correctness by construction
- Heterogeneity
 - synchronous and asynchronous execution
 - event- and data-driven computation
 - centralised and distributed implementation

Outline

- Motivation
- BIP and the Glue
- Synthesizing glue operators
- Design flow

Component design by refinement

Three layers:

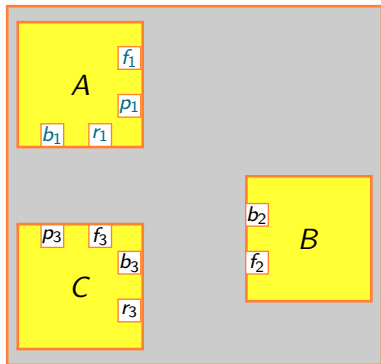
- 1 Component behaviour
- 2 Coordination
- 3 Data transfer



Component design by refinement

Three layers:

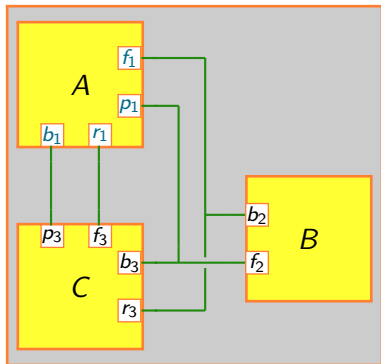
- 1 Component behaviour
- 2 Coordination
- 3 Data transfer



Component design by refinement

Three layers:

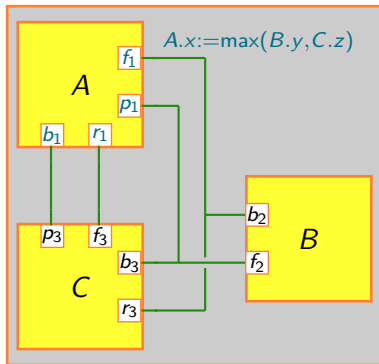
- 1 Component behaviour
- 2 Coordination
- 3 Data transfer



Component design by refinement

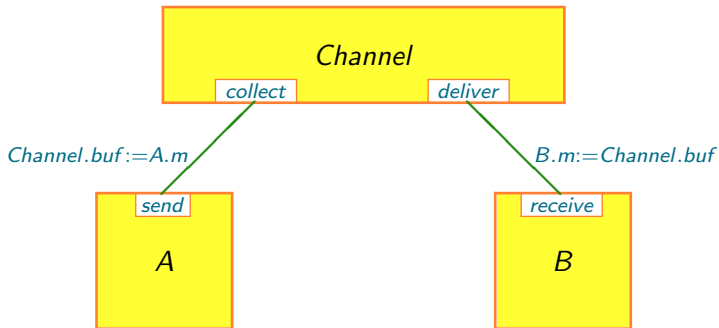
Three layers:

- 1 Component behaviour
- 2 Coordination
- 3 Data transfer



Unbuffered synchronous communication

(Not to confuse with synchronous *execution!*)



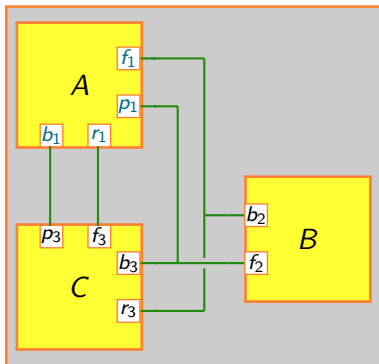
A sends a message m to B:

- Two synchronisations with the channel
- Each synchronisation allows a data transfer
- An explicit model of the channel behaviour

Scope of the basic BIP model

Three layers:

- 1 Component behaviour
- 2 Coordination
- 3 Data transfer



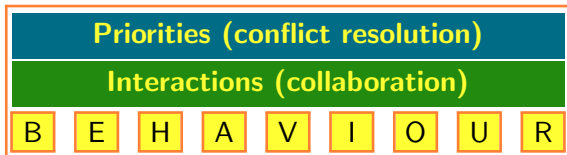
Interesting results already at this level, e.g.

- Analysis of synchronisation deadlocks

- S. Bensalem, M. Bozga, J. Sifakis, T.-H. Nguyen. *D-Finder: A Tool for Compositional Deadlock Detection and Verification*. [CAV'09]

- Synthesis of glue for safety properties

Basic model of BIP

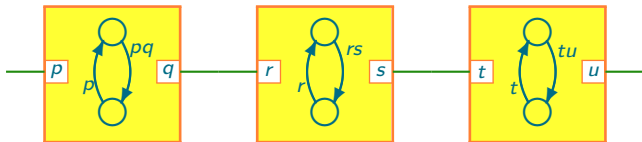


Layered component model

- **Behaviour** — labelled transition systems with disjoint sets of ports
- **Interaction** — set of interactions (interaction = set of ports)
- **Priorities** — strict partial order on interactions

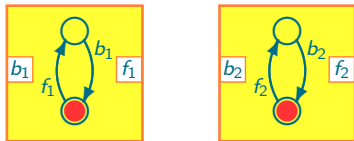
BIP examples

Modulo-8 counter:



Interactions: $\{p, pqr, pqrst, pqrstu\}$.

Mutual exclusion:



Interactions: $\{b_1, f_1, b_2, f_2\}$

Priority: $b_1 < f_2, b_2 < f_1$.

Glue semantics in BIP: Solid

$B_i = (Q_i, P_i, \rightarrow_i, \uparrow_i)$: P_i pairwise disjoint, $P = \bigcup_i P_i$

$$\rightarrow \subseteq Q \times 2^P \times Q$$

$$\uparrow \subseteq Q \times P \text{ such that } (\exists a \in 2^P : p \in a \wedge q \xrightarrow{a}) \Rightarrow q \uparrow p$$

Interaction model: $\gamma \subseteq 2^P$ — set of allowed interactions

$$\frac{\left\{ q_i \xrightarrow{a \cap P_i} q'_i \mid i \in [1, n], a \cap P_i \neq \emptyset \right\}}{q_1 \dots q_n \xrightarrow{a} \tilde{q}_1 \dots \tilde{q}_n} \text{ for each } a \in \gamma,$$

where \tilde{q}_i denotes q'_i if $a \cap P_i \neq \emptyset$, and q_i otherwise.

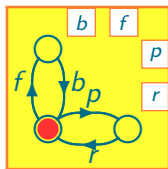
Priority model: $\prec \subseteq 2^P \times 2^P$ — strict partial order

$$\frac{q \xrightarrow{a} q' \{q \not\prec a' \mid a \prec a'\}}{q \xrightarrow{a} \prec q'} \text{ for each } a \in 2^P$$

Outline

- Motivation
- BIP and the Glue
- Synthesizing glue operators
- Design flow

Connector synthesis



Mutual preemption:

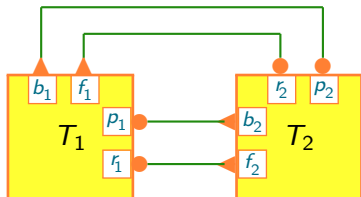
- 1 A running task is preempted, when the other one begins computation.
- 2 A preempted task resumes computation, when the other one finishes.

$$true \Rightarrow b_1 \vee f_1 \vee b_2 \vee f_2$$

$$p_1 \Rightarrow b_2 \quad p_2 \Rightarrow b_1$$

$$r_1 \Rightarrow f_2 \quad r_2 \Rightarrow f_1$$

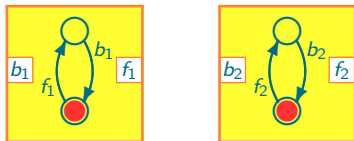
Mutual exclusion?..



$$\{b_1, b_2, b_1p_2, b_2p_1, f_1, f_2, f_1r_2, f_2r_1\}$$

S. Blidze, J. Sifakis. Causal semantics for the algebra of connectors. In *Formal Methods in System Design*, 2010.

Mutual exclusion (design front-end)



- 1 B_1 can enter the critical state if B_2 is in the non-critical one or leaves the critical state simultaneously

$$\text{fire}(b_1) \Rightarrow \neg \text{active}(f_2) \vee \text{fire}(f_2)$$

- 2 Idem for B_2 :

$$\text{fire}(b_2) \Rightarrow \neg \text{active}(f_1) \vee \text{fire}(f_1)$$

- 3 B_1 and B_2 cannot enter the critical state simultaneously

$$\neg (\text{fire}(b_1) \wedge \text{fire}(b_2))$$

Mutual exclusion (semantic back-end)

Notation: For a port $p \in P$, let p and \dot{p} — boolean *activation* and *firing* variables

Constraints:

$(\dot{b}_1 \Rightarrow \overline{\dot{f}_2} \vee \dot{f}_2) \wedge (\dot{b}_2 \Rightarrow \overline{\dot{f}_1} \vee \dot{f}_1) \wedge \overline{\dot{b}_1 \dot{b}_2}$ — Mutual exclusion

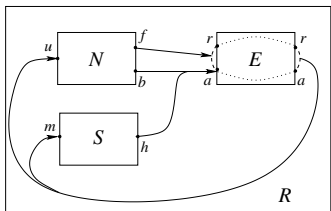
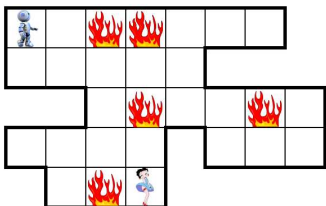
$\wedge (b_1 \vee f_1 \vee b_2 \vee f_2)$ — Progress

$\wedge \overline{\dot{f}_1 \dot{f}_2} \wedge (\dot{f}_1 \vee \dot{f}_2 \Rightarrow \overline{\dot{b}_1} \overline{\dot{b}_2})$ — “Internality” of finish

$= \overline{\dot{b}_1} \overline{\dot{b}_2} \dot{f}_1 \overline{\dot{f}_2} \vee \overline{\dot{b}_1} \overline{\dot{b}_2} \overline{\dot{f}_1} \dot{f}_2 \vee \dot{b}_1 \overline{\dot{b}_2} \overline{\dot{f}_1} \overline{\dot{f}_2} \overline{\dot{f}_2} \vee \overline{\dot{b}_1} \dot{b}_2 \overline{\dot{f}_1} \overline{\dot{f}_2} \overline{\dot{f}_1}$

$$\frac{q_1 \xrightarrow{f_1} q'_1}{q_1 q_2 \xrightarrow{f_1} q'_1 q_2}, \quad \frac{q_2 \xrightarrow{f_2} q'_2}{q_1 q_2 \xrightarrow{f_2} q_1 q'_2}, \quad \underbrace{\frac{q_1 \xrightarrow{b_1} q'_1 \quad q_2 \not\xrightarrow{f_2}}{q_1 q_2 \xrightarrow{b_1} q'_1 q_2}, \quad \frac{q_1 \not\xrightarrow{f_1} \quad q_2 \xrightarrow{b_2} q'_2}{q_1 q_2 \xrightarrow{b_2} q_1 q'_2}}_{\text{Priorities: } b_1 \prec f_2, b_2 \prec f_1}$$

Rescue robot (design front-end)



- 1 Must not advance and rotate at the same time: $\bar{a}\bar{r}$;
- 2 Must not leave the region: $b \Rightarrow \bar{a}$;
- 3 Must not drive into hot areas: $h \Rightarrow \bar{a}$;
- 4 Must stop, when objective is found: $f \Rightarrow \bar{a}\bar{r}$;
- 5 Must update navigation and sensor data on every move (advance or rotate): $\dot{a} \vee \dot{r} \Rightarrow \dot{u}\dot{m}$.

Rescue robot (semantic back-end)

$$\begin{aligned}
 & \bar{a} \bar{r} \wedge (b \Rightarrow \bar{a}) \wedge (h \Rightarrow \bar{a}) \wedge (f \Rightarrow \bar{a} \bar{r}) \wedge (\dot{a} \vee \dot{r} \Rightarrow \dot{u} \dot{m}) \quad \text{— Safety} \\
 & \wedge (\dot{a} \vee \dot{r} \vee \dot{u} \vee \dot{m}) \wedge \bar{h} \bar{b} \bar{f} \quad \text{— Progress} \\
 & = \left(\bar{a} \bar{r} \dot{u} \bar{m} \vee \bar{a} \bar{r} \dot{u} \dot{m} \vee \bar{a} \bar{r} \bar{u} \dot{m} \vee \bar{a} \dot{r} \bar{f} \dot{u} \dot{m} \vee \dot{a} \bar{r} \bar{b} \bar{h} \bar{f} \dot{u} \dot{m} \right) \wedge \bar{h} \bar{b} \bar{f}
 \end{aligned}$$

$$\frac{q_n \xrightarrow{u} q'_n}{q_e q_s q_n \xrightarrow{u} q_e q_s q'_n}, \quad \frac{q_s \xrightarrow{m} q'_s \quad q_n \xrightarrow{u} q'_n}{q_e q_s q_n \xrightarrow{mu} q_e q'_s q'_n}, \quad \frac{q_s \xrightarrow{m} q'_s}{q_e q_s q_n \xrightarrow{m} q_e q'_s q_n},$$

$$\frac{q_e \xrightarrow{r} q'_e \quad q_s \xrightarrow{m} q'_s \quad q_n \xrightarrow{u} q'_n \quad q_n \not\ll h}{q_e q_s q_n \xrightarrow{rmu} q'_e q'_s q'_n},$$

$$\frac{q_e \xrightarrow{a} q'_e \quad q_s \xrightarrow{m} q'_s \quad q_n \xrightarrow{u} q'_n \quad q_s \not\ll h \quad q_n \not\ll b \quad q_n \not\ll f}{q_e q_s q_n \xrightarrow{amu} q'_e q'_s q'_n}.$$

General case

Constraints: $\mathbb{B}[P, \dot{P}]$ with an axiom $\dot{p} \Rightarrow p$

SOS rules:

$$\frac{\left\{ B_i : q_i \xrightarrow{a_i} q'_i \right\}_{i \in I} \quad \left\{ B_j : q_j \uparrow b_j \right\}_{j \in J} \quad \left\{ B_k : q_k \not\uparrow c_s \mid s \in L_k \right\}_{k \in K}}{gl(B_1, \dots, B_n) : q_1 \dots q_n \xrightarrow{a} \tilde{q}_1 \dots \tilde{q}_n}$$

Theorem

Constraint glues and SOS glues are equivalent.

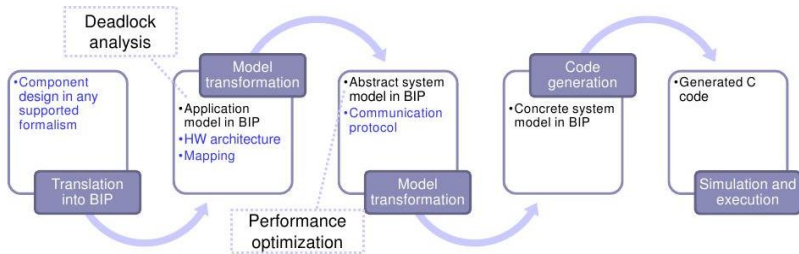
Outline

- Motivation
- BIP and the Glue
- Synthesizing glue operators
- Design flow

Design flow

- 1 Choice of the functionalities to be realized by *sequential atomic components*.
- 2 Independent design of sequential atomic components.
- 3 Specification of *state* safety properties to be satisfied by the system.
- 4 Automatic glue operator and connector synthesis. This implies that the underlying state safety properties are satisfied *by construction*.

Existing BIP design flow



- Models & information at different design stages
 - In light blue – provided by the designer
 - In black – generated by automatic transformation tools

<http://www.slideshare.net/sbliudze/bip-design-flow>

<http://www-verimag.imag.fr/The-BIP-Design-Flow.html>

Conclusion



We have

- Taken BIP one step closer to something
 - Solid — by improving semantics of hierarchical composition
 - Light-weight — by isolating designers from low-level details
- Through separation of concerns, reduced a very hard problem of synthesizing controllers to a tractable one.
- Given a natural boolean characterisation of glue through constraints \Rightarrow symbolic manipulation with BDDs.

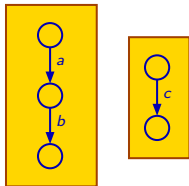
Thank you for your attention!

SOS operator example

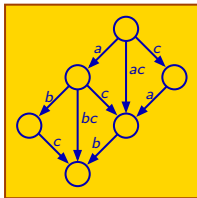
Glue operator g defined by the following rules

$$\left\{ \frac{q_1 \xrightarrow{a} q'_1}{q_1 q_2 \xrightarrow{a} q'_1 q_2}, \frac{q_1 \xrightarrow{a} q'_1 \quad q_2 \xrightarrow{c} q'_2}{q_1 q_2 \xrightarrow{ac} q'_1 q'_2}, \frac{q_1 \xrightarrow{b} q'_1 \quad q_2 \xrightarrow{c} q'_2}{q_1 q_2 \xrightarrow{b} q'_1 q_2} \right\}$$

Behaviours
 B_1, B_2



Parallel product
 $B_1 \parallel B_2$



Application of glue
 $g(B_1, B_2)$

