

# Modules for Service Component Architectures

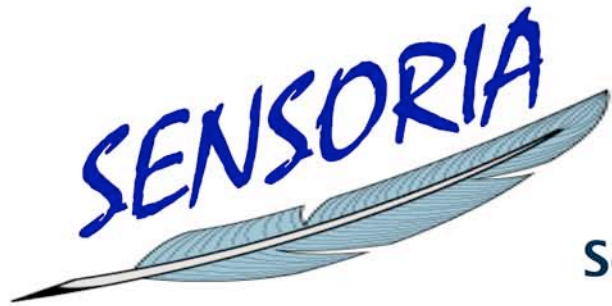
---

José L Fiadeiro  
Laura Bocchi



Antónia Lopes





An IST-FET Integrated Project **Sept05–Aug09**

## Software Engineering for Service-Oriented Overlay Computers

The aim of SENSORIA is to develop a novel comprehensive approach to the engineering of software systems for service-oriented overlay computers where foundational theories, techniques and methods are fully integrated in a pragmatic software engineering approach.

- WP1** Provide support for service-oriented modelling at high levels of abstraction, i.e. independently of the hosting middleware and hardware platforms, and the languages in which services are programmed.
- T1.1** **Develop a prototype language for service description and composition including syntax and mathematical semantics.**

- Web-Services (Alonso, Benatallah, Casati, ...)
- **ORC** (Misra)
- **Architectural** modelling of mobile and distributed systems
- **SCA – Service Component Architecture**  
(BEA, IBM, Interface21, IONA, Oracle, SAP, Siebel, Sybase)  
*“a specification that [...] aims to simplify the creation and integration of business applications built using a Service Oriented Architecture”.*

SRML provides primitives for modelling **composite services** understood as services whose business logic involves a number of **interactions** among more elementary **service components** as well the invocation of services provided by **external parties**.



modelling ≠  
programming

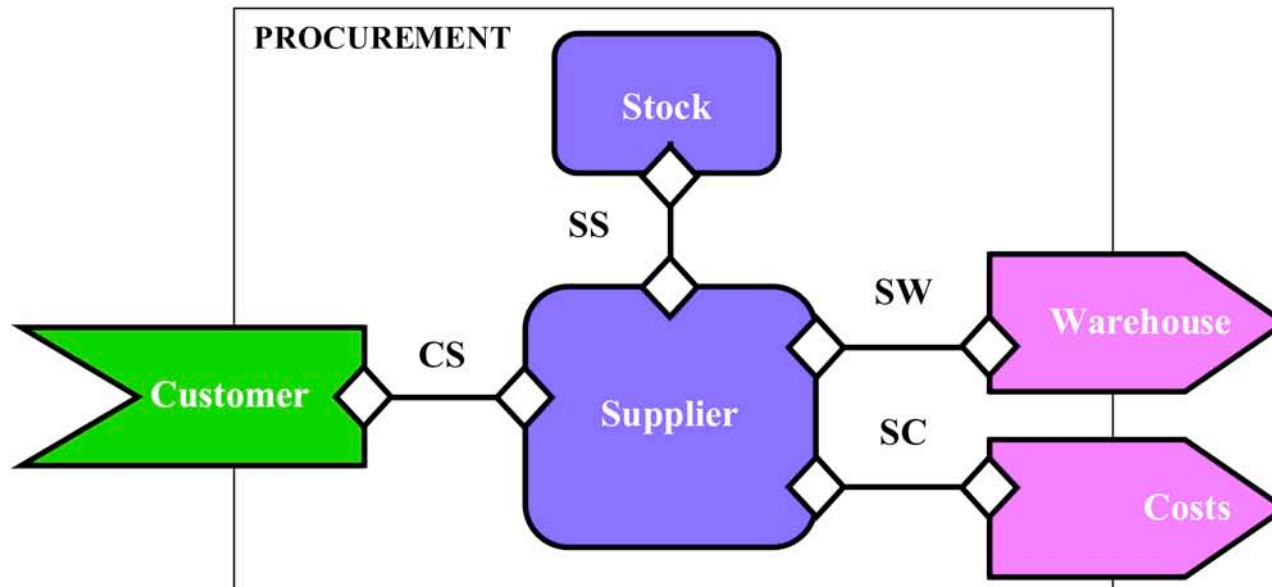
SRML provides primitives for modelling **composite services** understood as services whose business logic involves a number of **interactions** among more elementary **service components** as well the invocation of services provided by **external parties**.

**Interactions** are supported on the basis of **service interfaces** defined in a way that is independent of the hardware platform, the operating system, the hosting middleware and the programming language used to implement the service.

The modelling primitives have a **mathematical semantics** that supports **analysis of correctness** understood in terms of properties of **stateful interactions** (not pre/post conditions) maintained with external parties and **service-level agreements**.

A composite service is modelled as a **module** consisting of:

- At most one **Provides-interface**, specified as a Business Protocol;
- A number of **Requires-interfaces**, specified as Business Protocols;
- A number of **Components**, specified as Business Roles;
- **Wires** connecting them, specified as Interaction Protocols.



# A business protocol

the behaviour  
**required** of a  
warehouse

**BUSINESS PROTOCOL** Warehouse **is**

## INTERACTIONS

**r&s** check&lock

🔔 which:product, many:nat

✉ Reply:bool

**snd** confirm

## BEHAVIOUR

**initially** check&lock🔔?

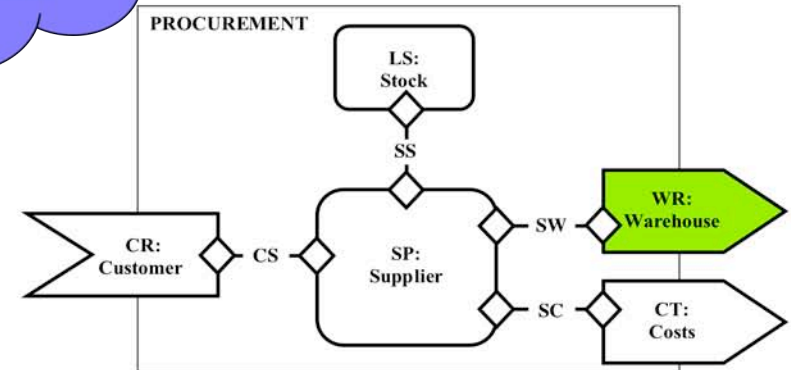
check&lock✉!  $\wedge$  check&lock.Reply

⊃ alertDate🔔!  $\wedge$  alertDate.Interval=3  $\wedge$  alertDate.Ref="goods"

check&lock💣! ⊃ alertDate✉?  $\wedge$  alertDate.Ref="goods"

check&lock⌚ ⊃ (check&lock✓? **ensures** confirm🔔!)

check&lock✓? ⊃ (check&lock✚? **exceptif** confirm🔔!)



# A business protocol

the behaviour  
**required** of a  
warehouse

**BUSINESS PROTOCOL** Warehouse **is**

## INTERACTIONS

**r&s** check&lock

🔔 which:product, many:nat

✉ Reply:bool

**snd** confirm

## BEHAVIOUR

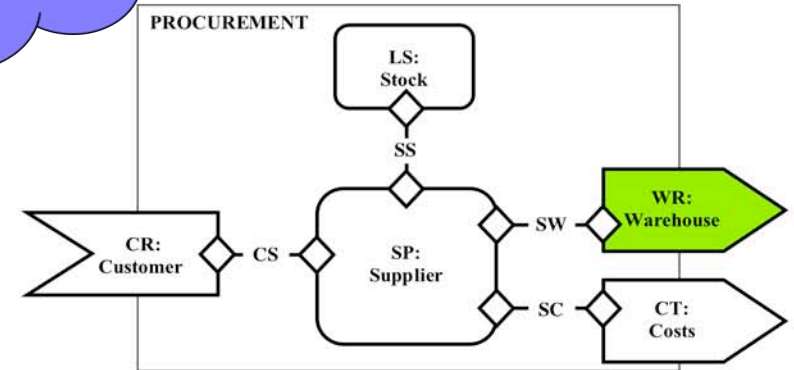
**initially** check&lock🔔?

check&lock✉!  $\wedge$  check&lock.R  
     $\supset$  alertDate🔔!  $\wedge$  alertDate.

check&lock💣!  $\supset$  alertDate✉?

check&lock🕒  $\supset$  (check&lock✓? e

check&lock✓?  $\supset$  (check&lock✚? exce



the warehouse is  
required to **accept**  
**request** for **check&lock**  
at the start of a new  
session



# A business protocol

the behaviour  
**required** of a  
warehouse

**BUSINESS PROTOCOL** Warehouse **is**

## INTERACTIONS

**r&s** check&lock

🔔 which:product, many:nat

✉ Reply:bool

**snd** confirm

## BEHAVIOUR

**initially** check&lock🔔?

check&lock✉!  $\wedge$  check&lock.Reply

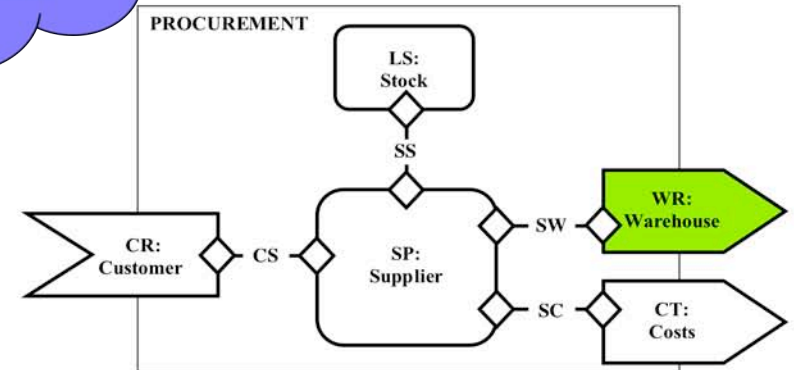
⊃ alertDate🔔!  $\wedge$  alertDate.Interval=3  $\wedge$  alertDate.Ref="goods"

check&lock💣! ⊃ alertDate🔔?  $\wedge$  alertDate.Interval=3  $\wedge$  alertDate.Ref="goods"

check&lock⌚ ⊃ (check&lock.Reply)

check&lock✓? ⊃

when the warehouse **issues** a  
positive **reply** to **check&lock**, it  
**sets** an **alertDate** with an interval  
of 3 days and reference "goods"



# A business protocol

the behaviour  
**required** of a  
warehouse

**BUSINESS PROTOCOL** Warehouse **is**

## INTERACTIONS

**r&s** check&lock

🔔 which:product, many:nat

✉ Reply:bool

**snd** confirm

## BEHAVIOUR

**initially** check&lock🔔?

check&lock✉!  $\wedge$  check&lock.Reply

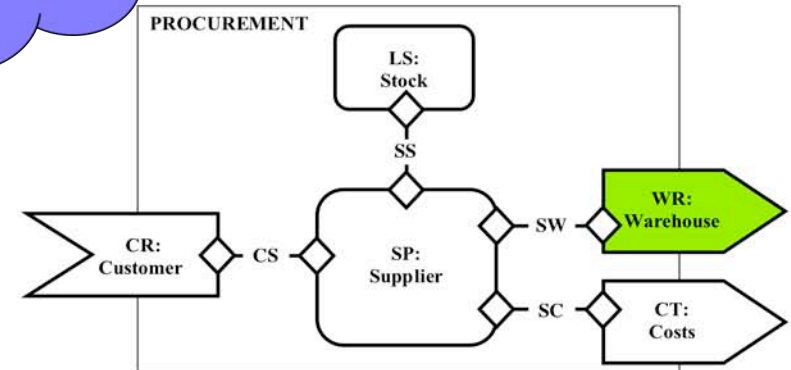
▷ alertDate🔔!  $\wedge$  alertDate.Interval=3  $\wedge$  alertDate.Ref="goods"

check&lock💣! ▷ alertDate✉?  $\wedge$  alertDate.Ref="goods"

check&lock🕒 ▷

check&lock✓? ▷

the **deadline** associated with  
**check&lock** is the reception of a  
reply to an **alertDate** with  
reference "goods"



# A business protocol

the behaviour  
**required** of a  
warehouse

**BUSINESS PROTOCOL** Warehouse **is**

## INTERACTIONS

**r&s** check&lock

🔔 which:product, many:nat

✉ Reply:bool

**snd** confirm

## BEHAVIOUR

**initially** check&lock🔔?

check&lock✉!  $\wedge$  check&lock.Reply

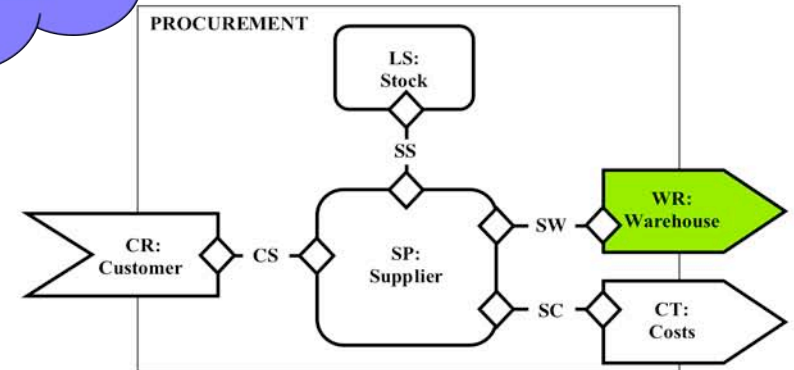
⊃ alertDate🔔!  $\wedge$  alertDate.Interval=3  $\wedge$  alertDate.Ref="goods"

check&lock💣! ⊃ alertDate✉?  $\wedge$  alertDate.Ref="goods"

check&lock⌚ ⊃ (check&lock✓? **ensures** confirm🔔!)

check&lock✓? ⊃ (select ...)

the **pledge** associated with  
**check&lock** ensures that a  
**confirm** is issued after a **commit**  
to **check&lock** is received.



# A business protocol

the behaviour  
**required** of a  
warehouse

**BUSINESS PROTOCOL** Warehouse **is**

## INTERACTIONS

**r&s** check&lock

🔔 which:product, many:nat

✉ Reply:bool

**snd** confirm

## BEHAVIOUR

**initially** check&lock🔔?

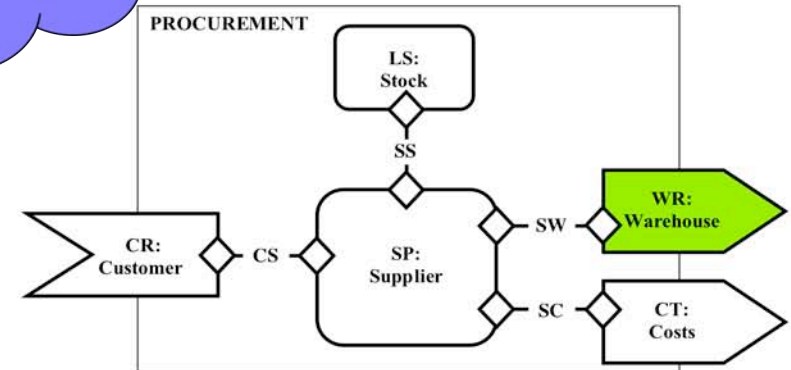
check&lock✉!  $\wedge$  check&lock.Reply

⊃ alertDate🔔!  $\wedge$  alertDate.Interval=3  $\wedge$  alertDate.Ref="goods"

check&lock💣! ⊃ alertDate✉?  $\wedge$  alertDate.Ref="goods"








check&lock⌚ ⊃ (check&lock✓? **ensures** confirm🔔!)

check&lock✓? ⊃ (check&lock✚? **exceptif** confirm🔔!)

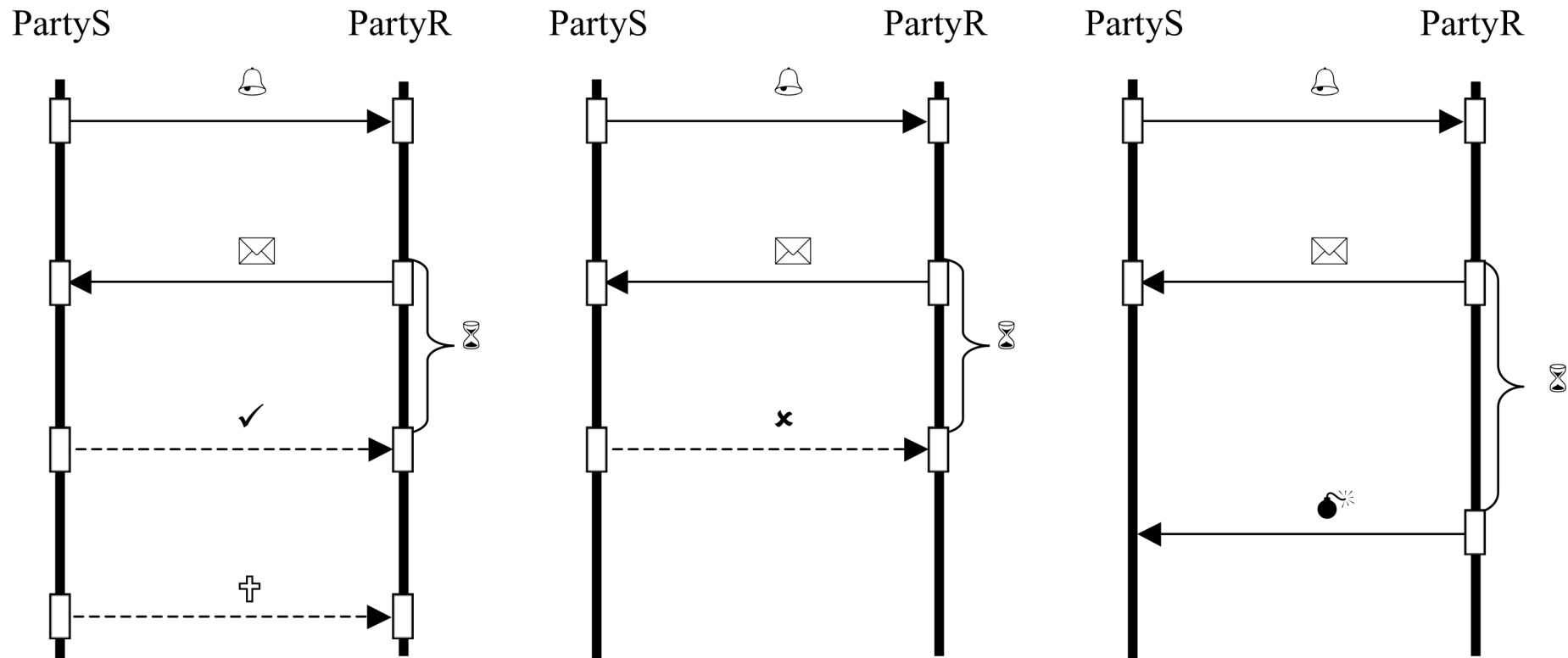


a commit to **check&lock**  
can be **revoked** any time  
before a **confirm** is issued.

## A catalogue of Interaction Description Primitives:

<i>interaction</i> 	The event of <b>initiating</b> <i>interaction</i> .
<i>interaction</i> 	The <b>reply-event</b> of <i>interaction</i>
<i>interaction</i> 	The <b>pledge</b> associated with <i>interaction</i> .
<i>interaction</i> 	The <b>timeout</b> of <i>interaction</i> , an event that signals that the pledge is no longer guaranteed to hold.
<i>interaction</i> 	The <b>commit-event</b> of <i>interaction</i> (the pledge is enforced).
<i>interaction</i> 	The <b>cancel-event</b> of <i>interaction</i> (the pledge is discarded).
<i>interaction</i> 	A <b>revoke</b> -event for <i>interaction</i> , which means cancelling the effects of <i>interaction</i> after having committed to it.

## A catalogue of Interaction Description Primitives:



Interactions are durative / stateful

A formal **model** and **logic** for reasoning about **interactions** are being developed (by João Abreu):

- Interactions are taken to be **durative** in the sense of involving a number of correlated events;
- Service behaviour is modelled in terms of **doubly labelled transition systems** (states and transitions are labelled);
- The logic is  **$\mu$ UCTL** developed at ISTI-CNR (Pisa), which subsumes both ACTL and CTL;
- Reasoning with  **$\mu$ UCTL** is supported by the model checker UMC.

# A business role

9  
|  
18

**BUSINESS ROLE** Supplier **is**  
**INTERACTIONS**

**ORCHESTRATION**

**local**

```
s:[0..8], inStock:bool, which:product,  
much:money, timeoutQuote:bool
```

**initialisation** s=0

**termination** s=8

**transition** Tquote

**triggeredBy** requestQuote🔔?

**guardedBy** s=0

**effects** which'=requestQuote.which

∧ much'=how(which')\*1.2

∧ inStock'=false

∧ timeoutQuote'=false

∧ s'=1

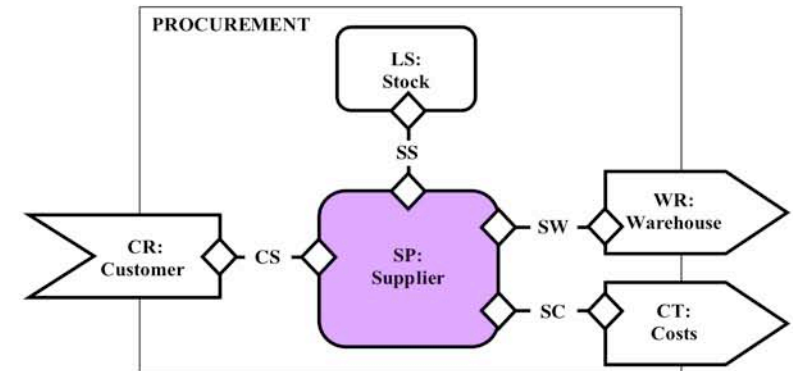
**sends** requestQuote📧!

∧ requestQuote.cost=much'

∧ alertDate🔔!

∧ alertDate.Ref="quote"

∧ alertDate.Interval=7





# A business role

**BUSINESS ROLE** Supplier is  
**INTERACTIONS**

**ORCHESTRATION**

**local**

```
s:[0..8], inStock:bool, which:product,  
much:money, timeoutQuote:bool
```

**initialisation** s=0

**termination** s=8

**transition** Tquote

**triggeredBy** requestQuote🔔?

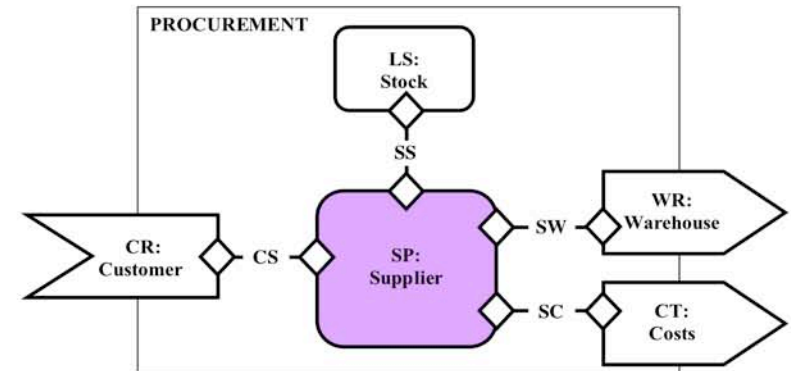
**guardedBy** s=0

**effects** which'=re

```
^ much'=how  
^ inStock'=fa  
^ timeoutQuote'=false  
^ s'=1
```

**sends** requestQuote☒!

```
^ requestQuote.cost=much'  
^ alertDate🔔!  
^ alertDate.Ref="quote"  
^ alertDate.Interval=7
```



a requestQuote is only accepted in an initial state (when s=0)

# A business role

**BUSINESS ROLE** Supplier **is**  
**INTERACTIONS**

**ORCHESTRATION**

**local**

```
s:[0..8], inStock:bool, which:product,  
much:money, timeoutQuote:bool
```

**initialisation** s=0

**termination** s=8

**transition** Tquote

**triggeredBy** requestQuote🔔?

**guardedBy** s=0

**effects** which'=requestQuote.which

∧ much'=how(which')\*1.2

∧ inStock'=1

∧ timeoutQuote'=1

∧ s'=1

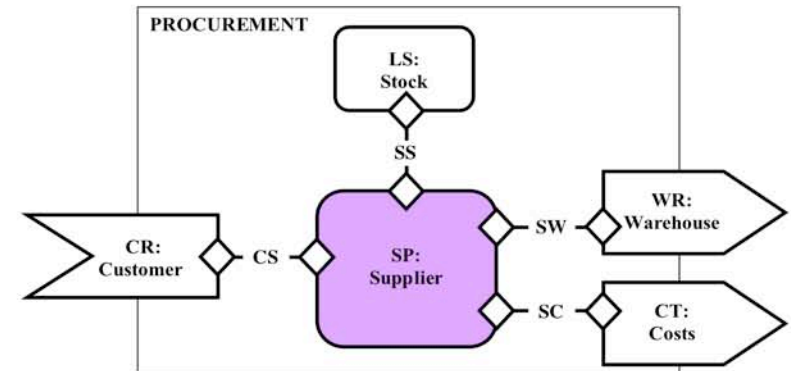
**sends** requestQuote

∧ requestQuote.cost

∧ alertDate🔔!

∧ alertDate.Ref="quote"

∧ alertDate.Interval=7



the value of **much** is set through a synchronous invocation of **how** on the requested product with a 20% overhead

# A business role

**BUSINESS ROLE Supplier is INTERACTIONS**

**ORCHESTRATION**

**local**

```
s:[0..8], inStock:bool, which:product,  
much:money, timeoutQuote:bool
```

**initialisation** s=0

**termination** s=8

**transition** Tquote

**triggeredBy** requestQuote🔔?

**guardedBy** s=0

**effects** which'=requestQuote.which

∧ much'=how(which')\*1.2

∧ inStock'=false

∧ timeoutQuote'=false

∧ s'=1

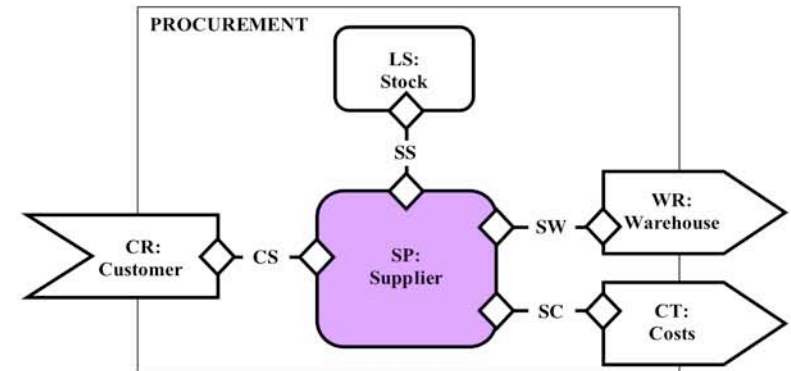
**sends** requestQuote📧!

∧ requestQuote.cost=much'

∧ alertDate🔔!

∧ alertDate.Ref="quote"

∧ alertDate.Interval=7



a reply to **requestQuote** is issued

# A business role

**BUSINESS ROLE Supplier is**  
**INTERACTIONS**

**ORCHESTRATION**

**local**

```
s:[0..8], inStock:bool, which:product,  
much:money, timeoutQuote:bool
```

**initialisation** s=0

**termination** s=8

**transition** Tquote

**triggeredBy** requestQuote🔔?

**guardedBy** s=0

**effects** which'=requestQuote.which

∧ much'=how(which')\*1.2

∧ inStock'=false

∧ timeoutQuote'=false

∧ s'=1

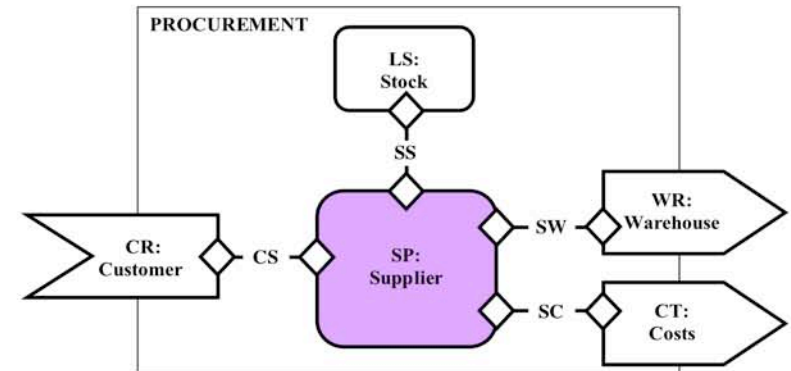
**sends** requestQuote📧!

∧ requestQuote.cost=much'

∧ alertDate🔔!

∧ alertDate.Ref="quote"

∧ alertDate.Interval=7



an **alertDate** is issued  
with a duration of 7 days  
and reference "quote".

# An interaction protocol

## INTERACTION PROTOCOL Custom1 is

### INTERACTIONS

```

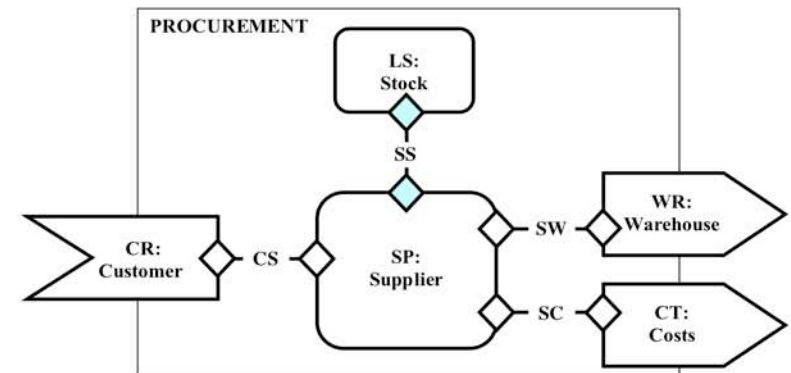
ask S1(product, nat):bool
tll S2(product, nat)
tll S3(product, nat)
rpl R1(product):nat
prf R2(product, nat)
    
```



### COORDINATION

$$S_1(p, n) = R_1(p) \geq n$$

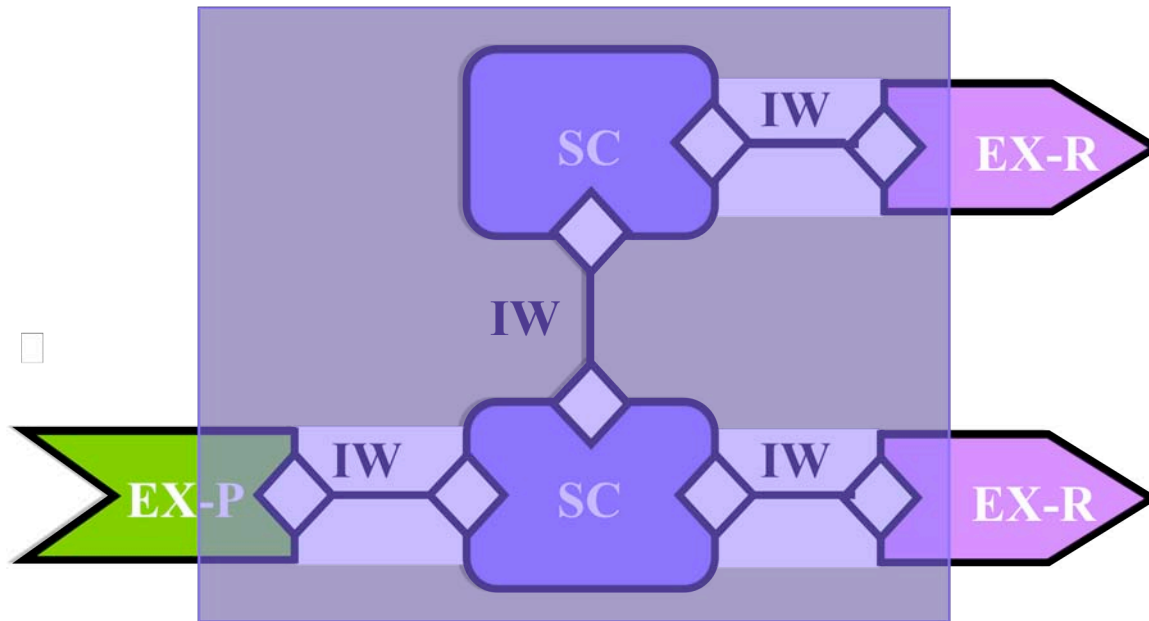
$$S_2(p, n) \supset R_2(p, R_1(p) + n)$$

$$R_1(p) \geq n \wedge S_3(p, n) \supset R_2(p, R_1(p) - n)$$

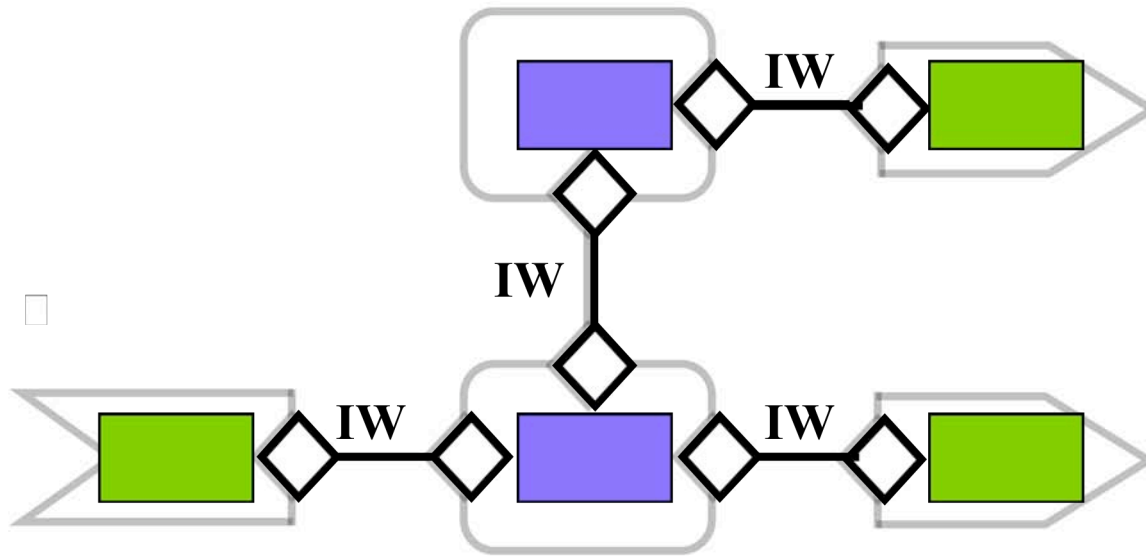
$$R_1(p) < n \supset \neg S_3(p, n)$$


SP Supplier		SS		LS Stock
<b>ask</b> checkStock <b>tll</b> incStock <b>tll</b> decStock	S <sub>1</sub> S <sub>2</sub> S <sub>3</sub>	Custom1	R <sub>1</sub> R <sub>2</sub>	<b>rpl</b> get <b>prf</b> set

# Modules as labelled graphs

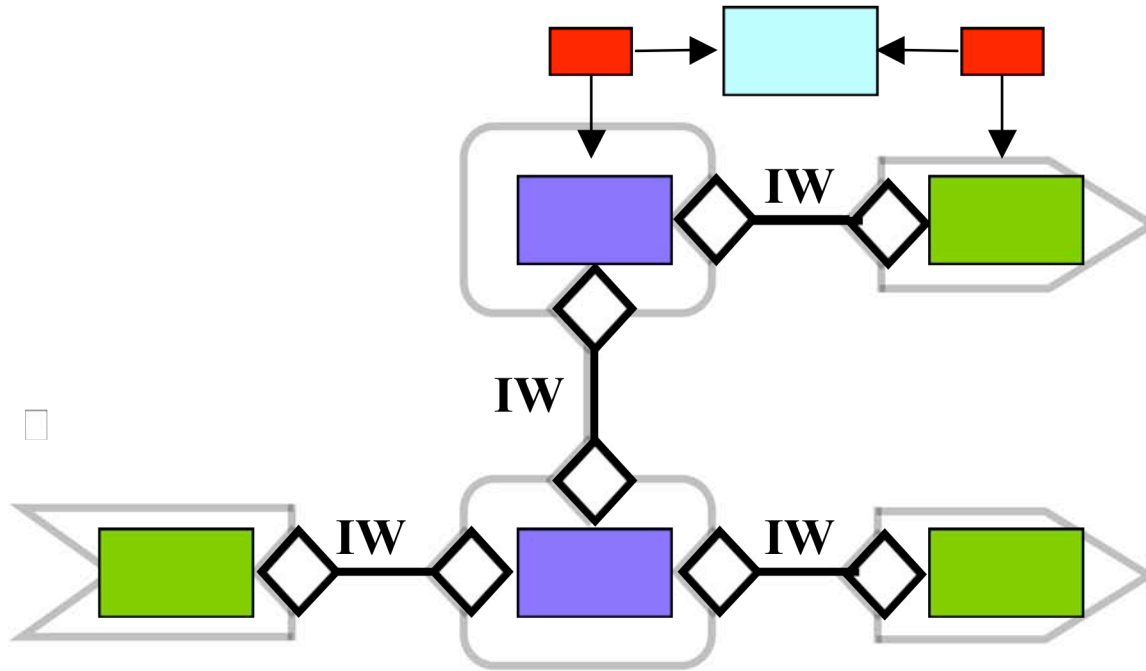


# Modules as labelled graphs



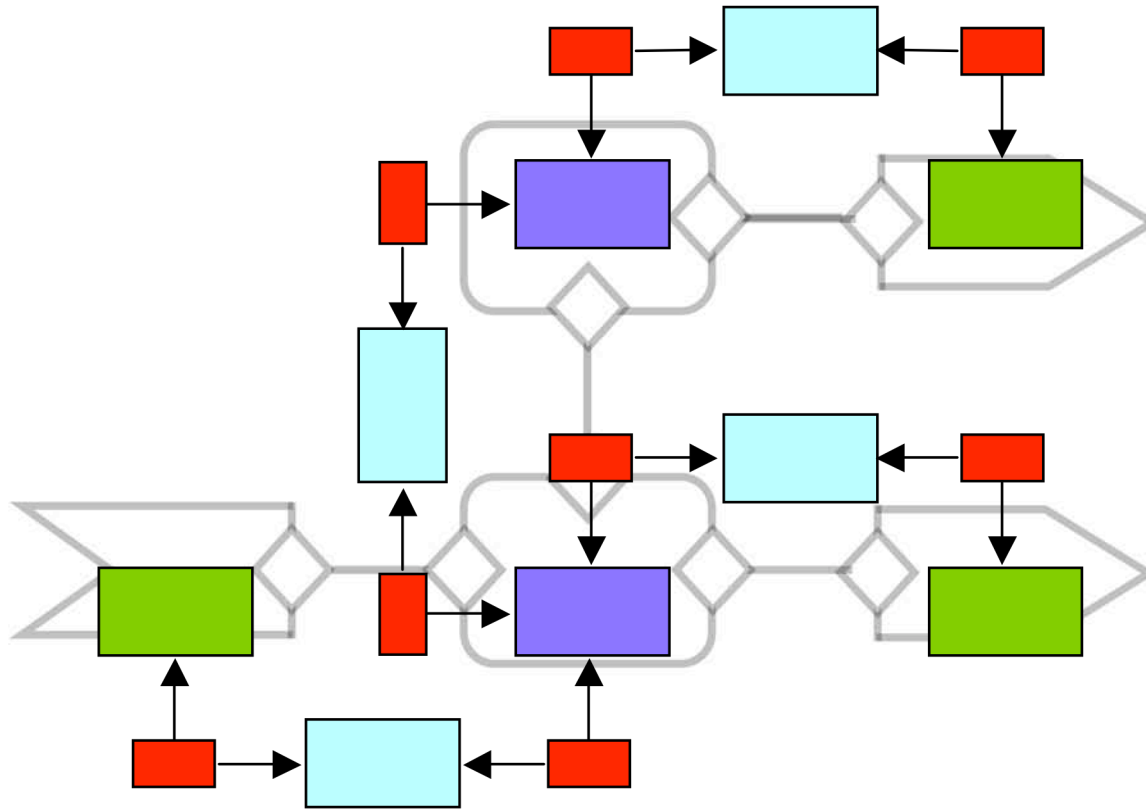
# Modules as labelled graphs

11  
—  
18





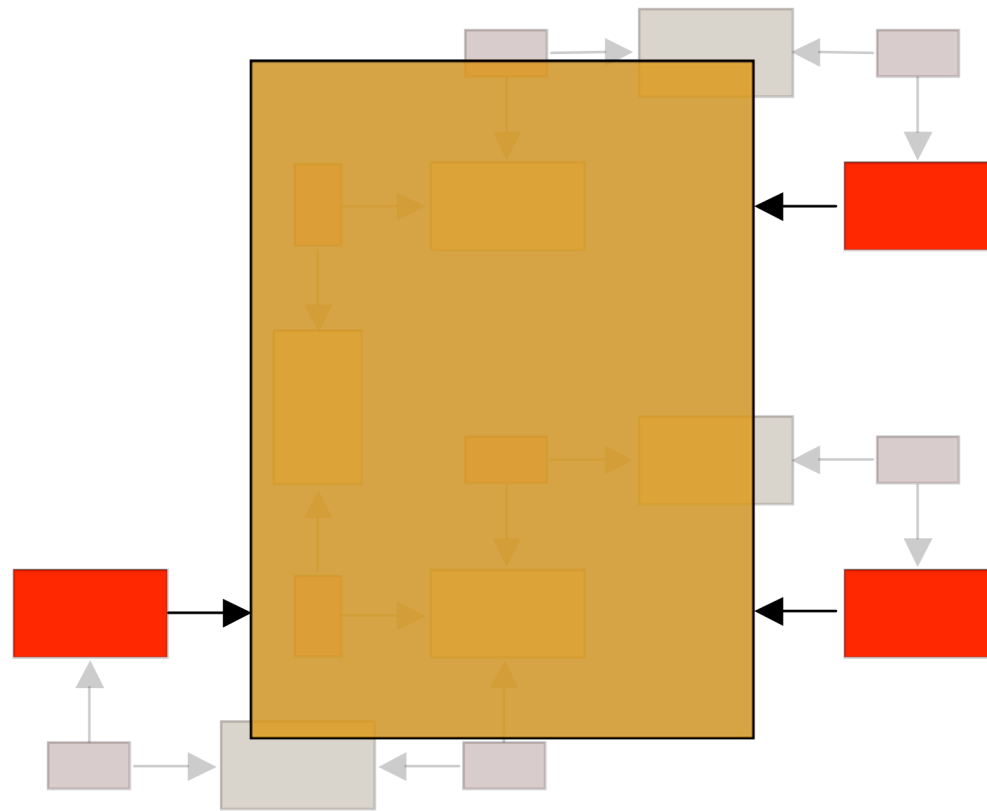
# Modules as labelled graphs



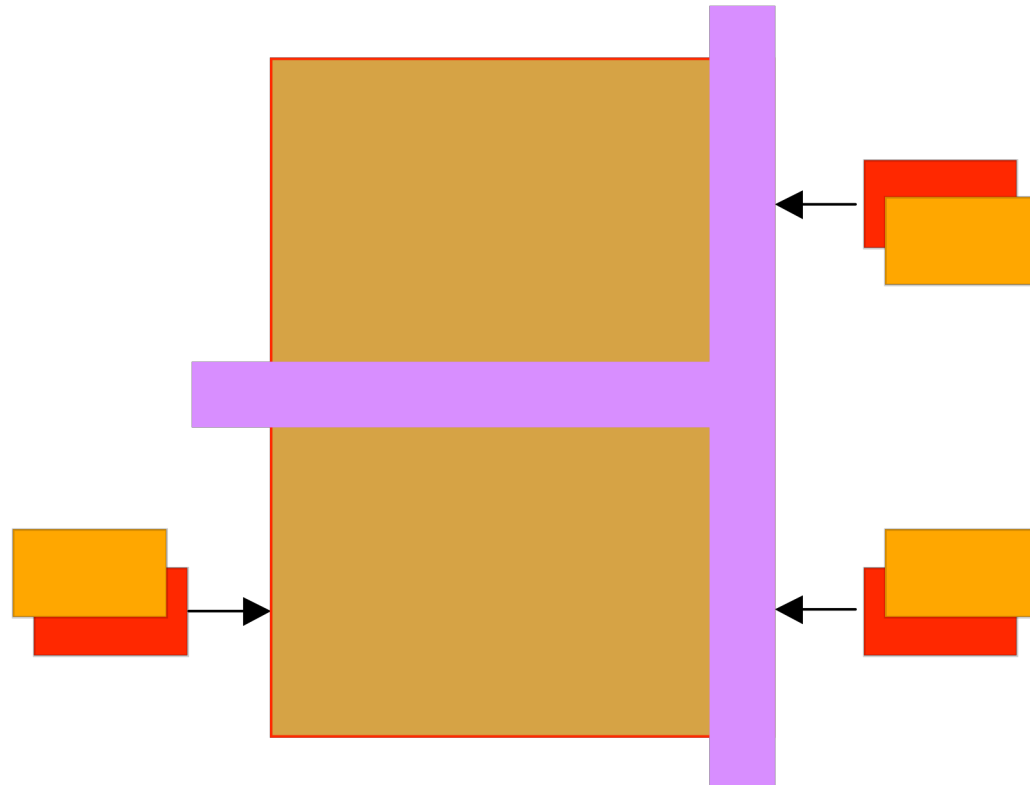
# The correctness property

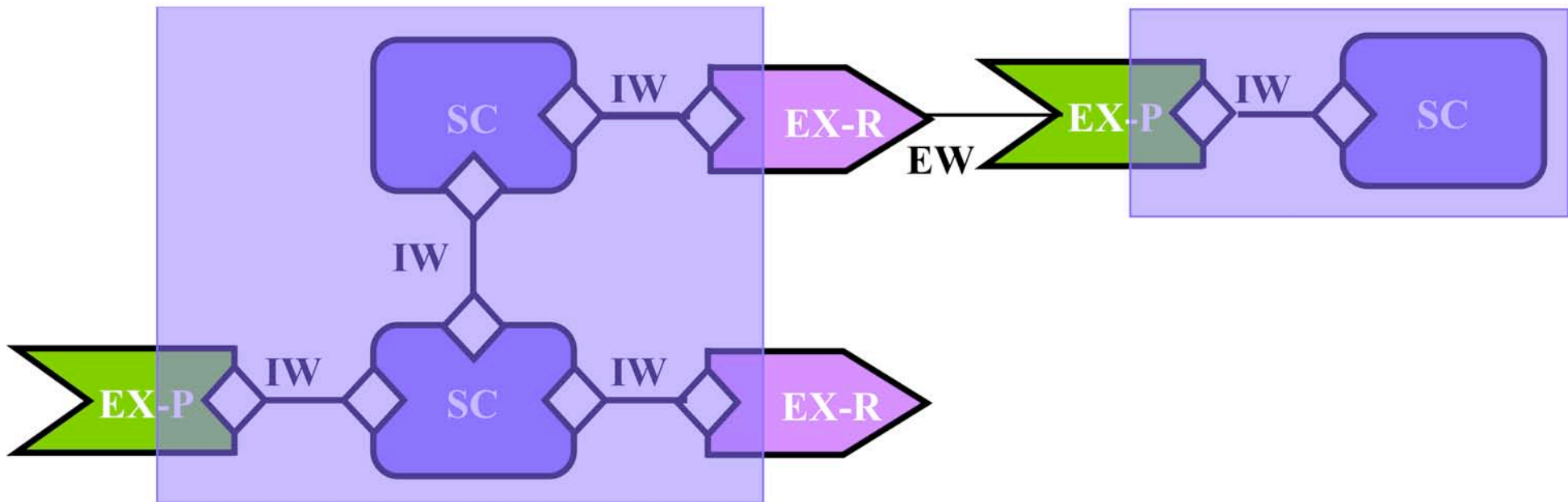
12  
—  
18

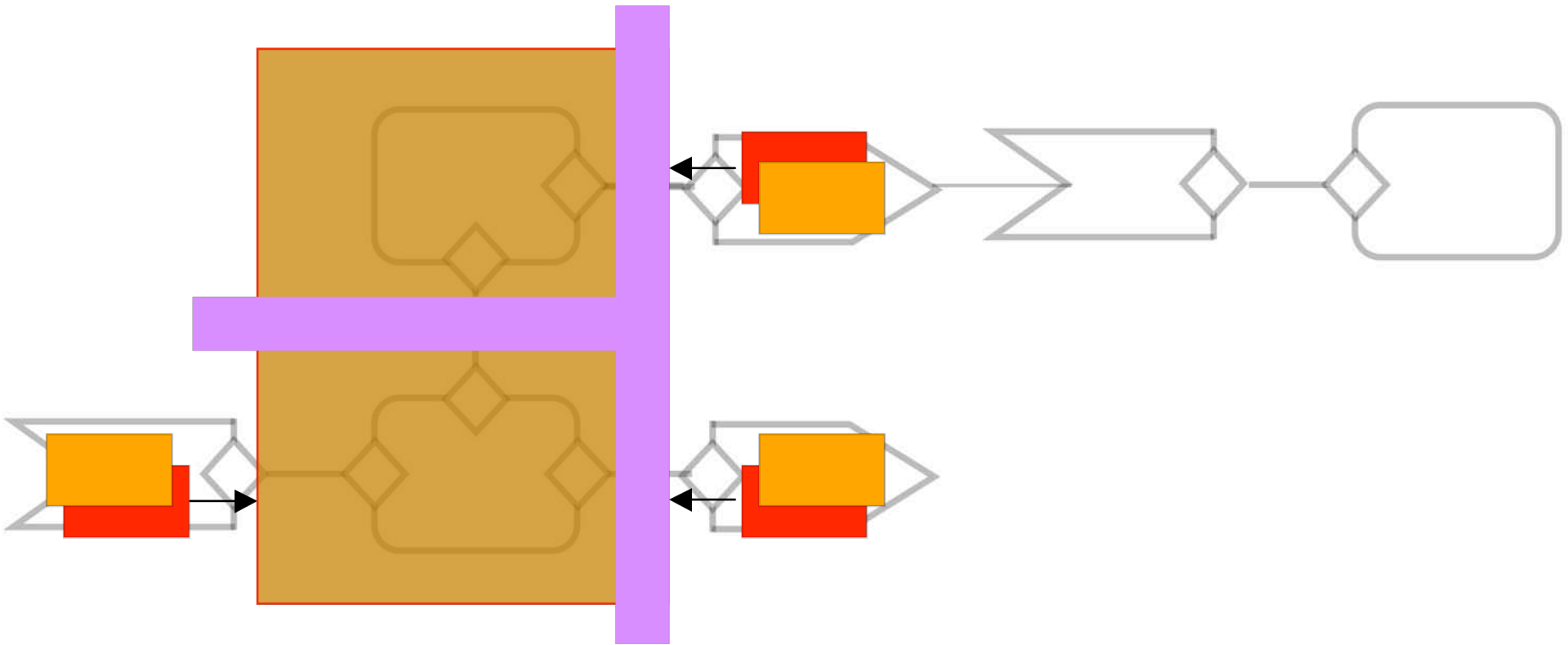
The **body** of the module

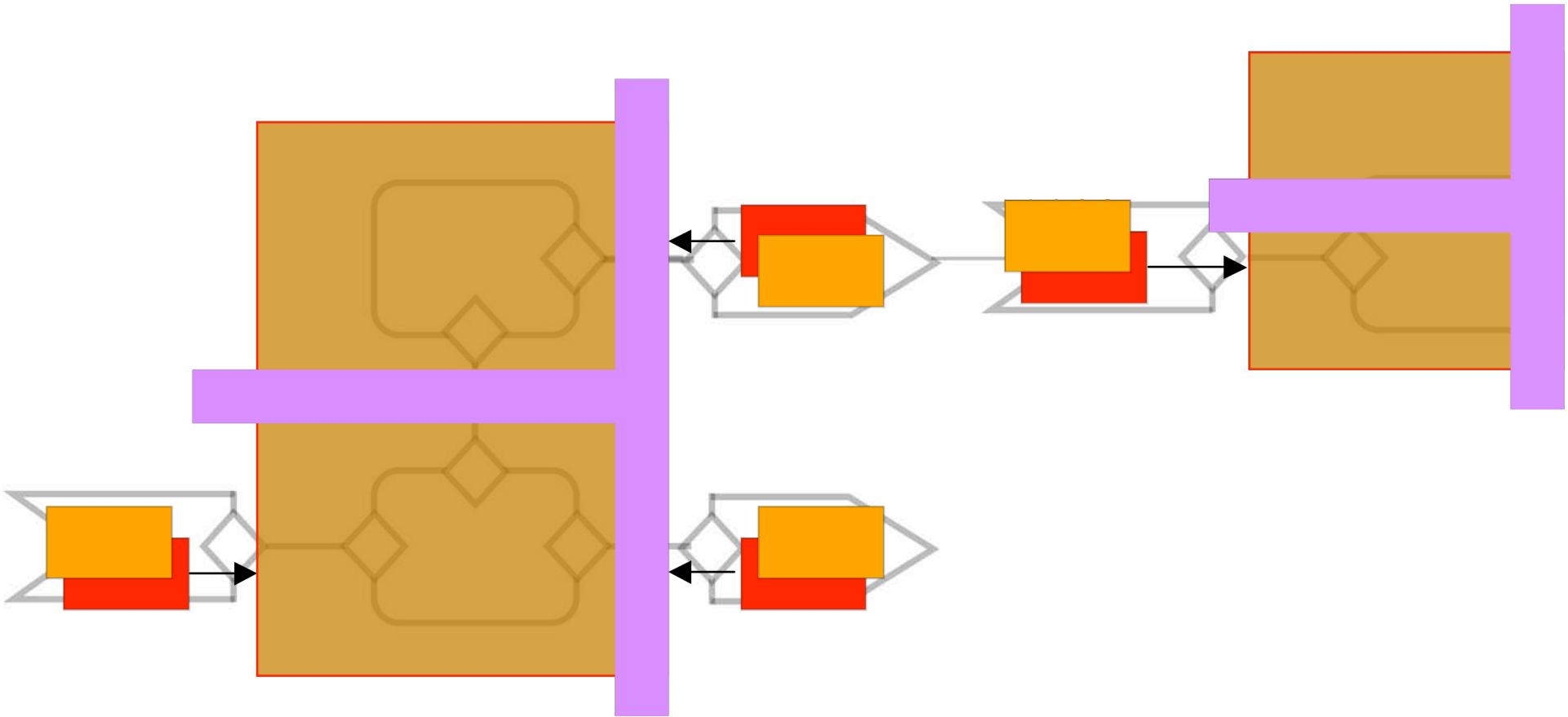


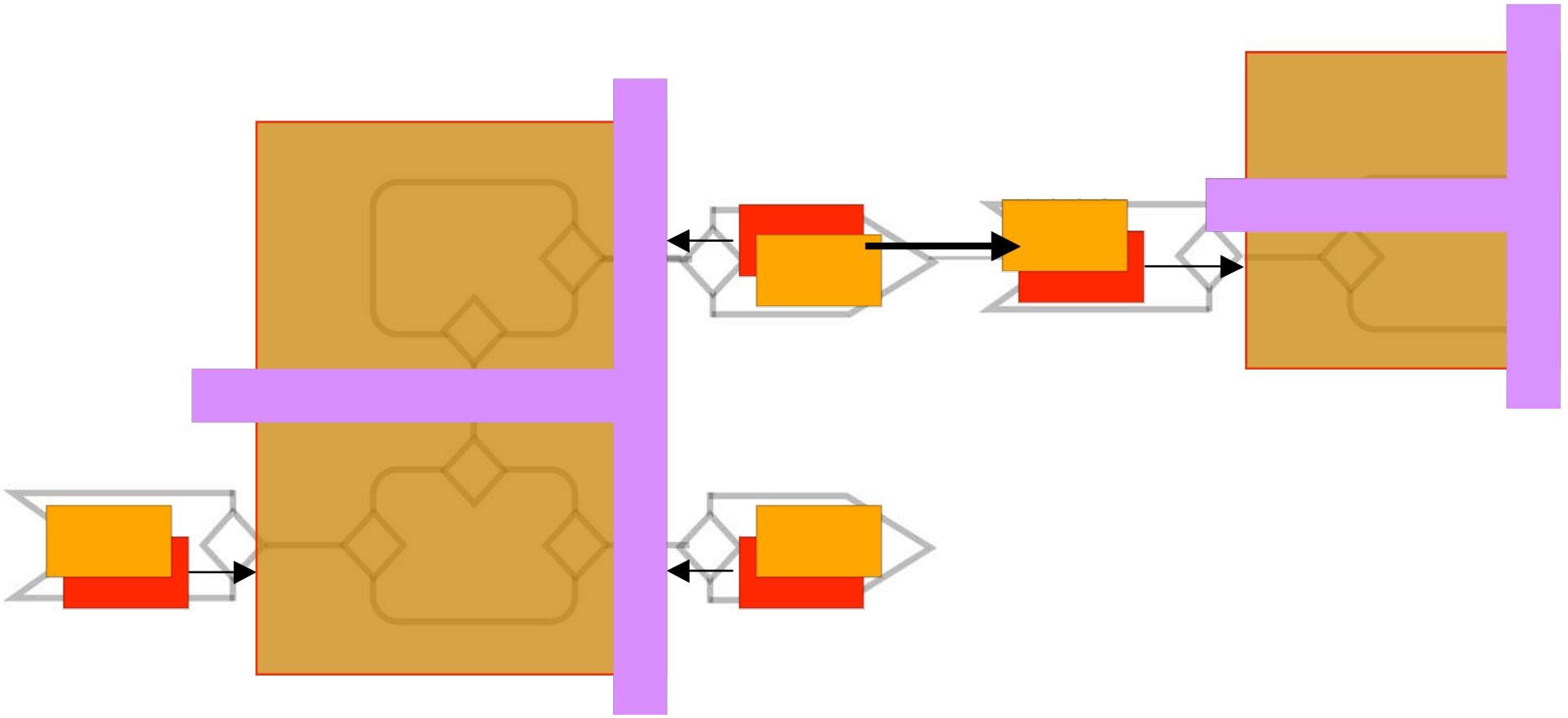
## Modules as judgements (clauses)

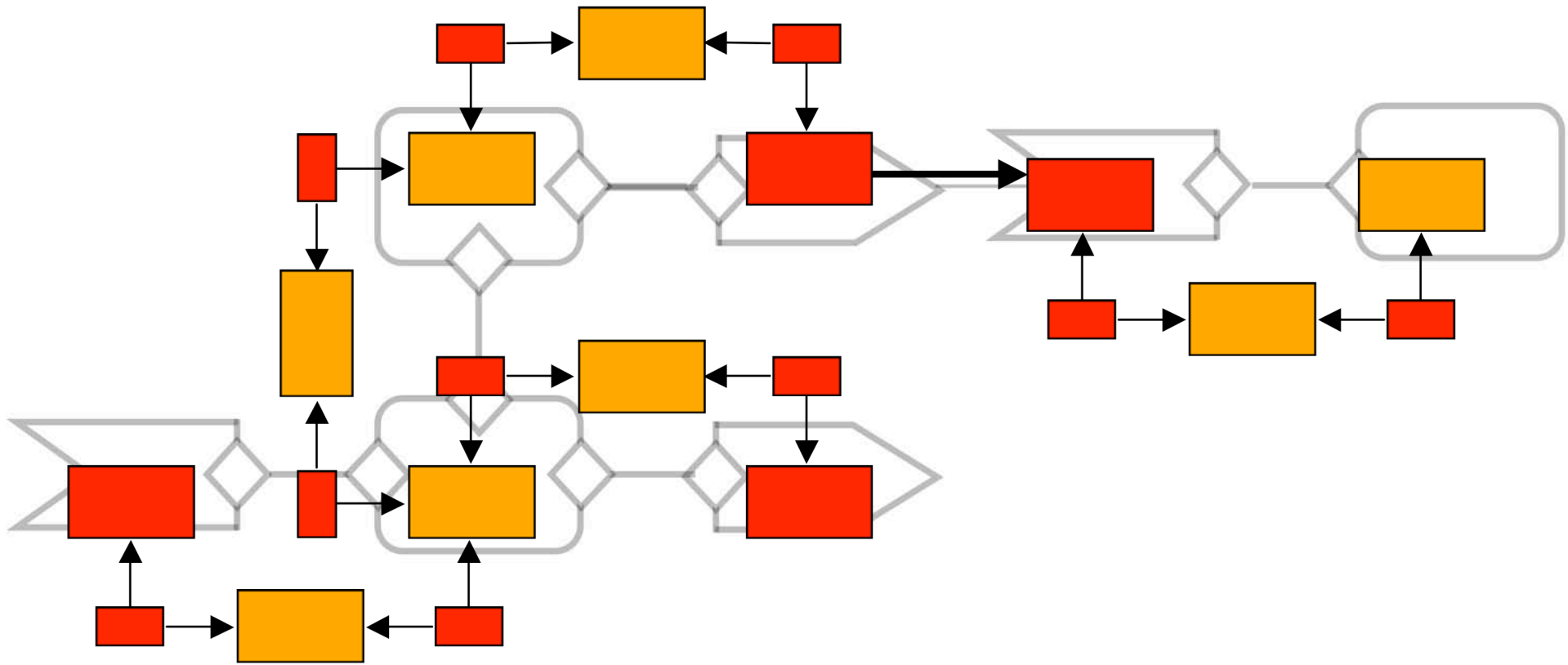




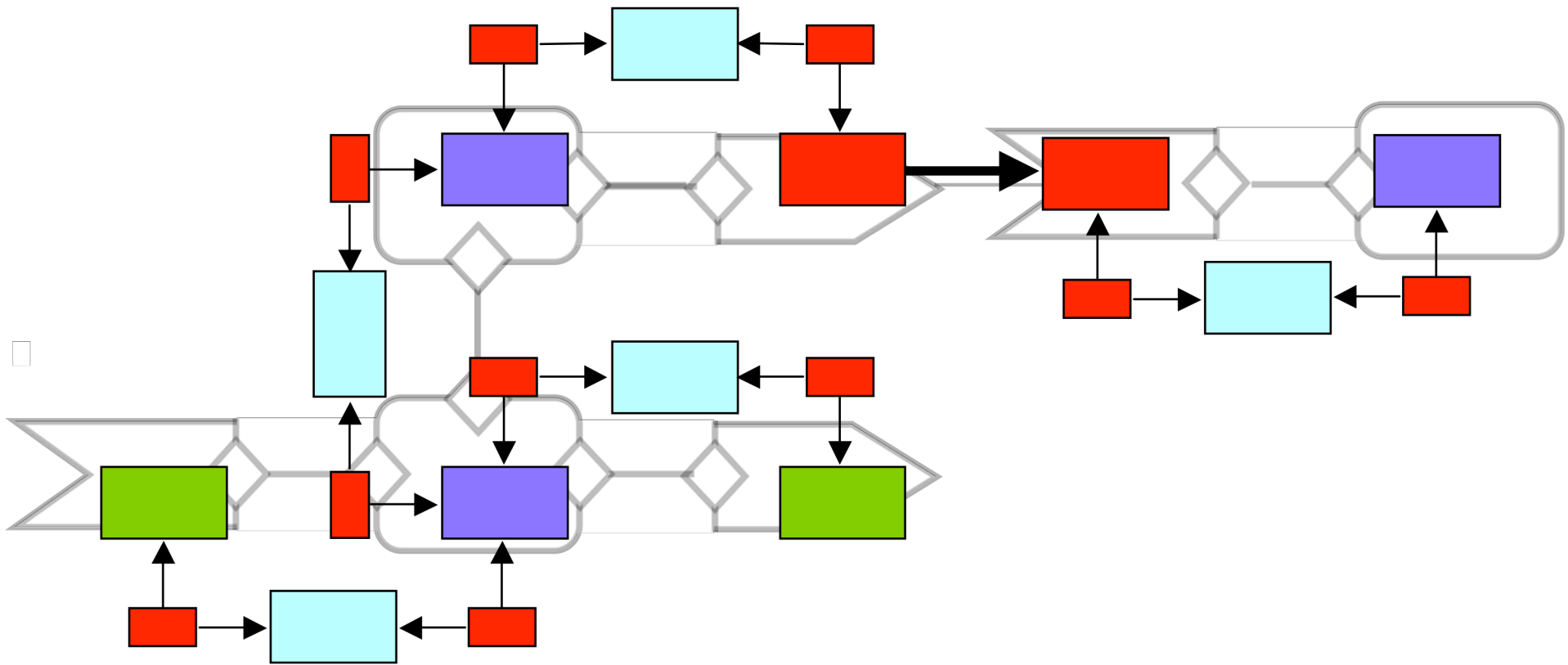


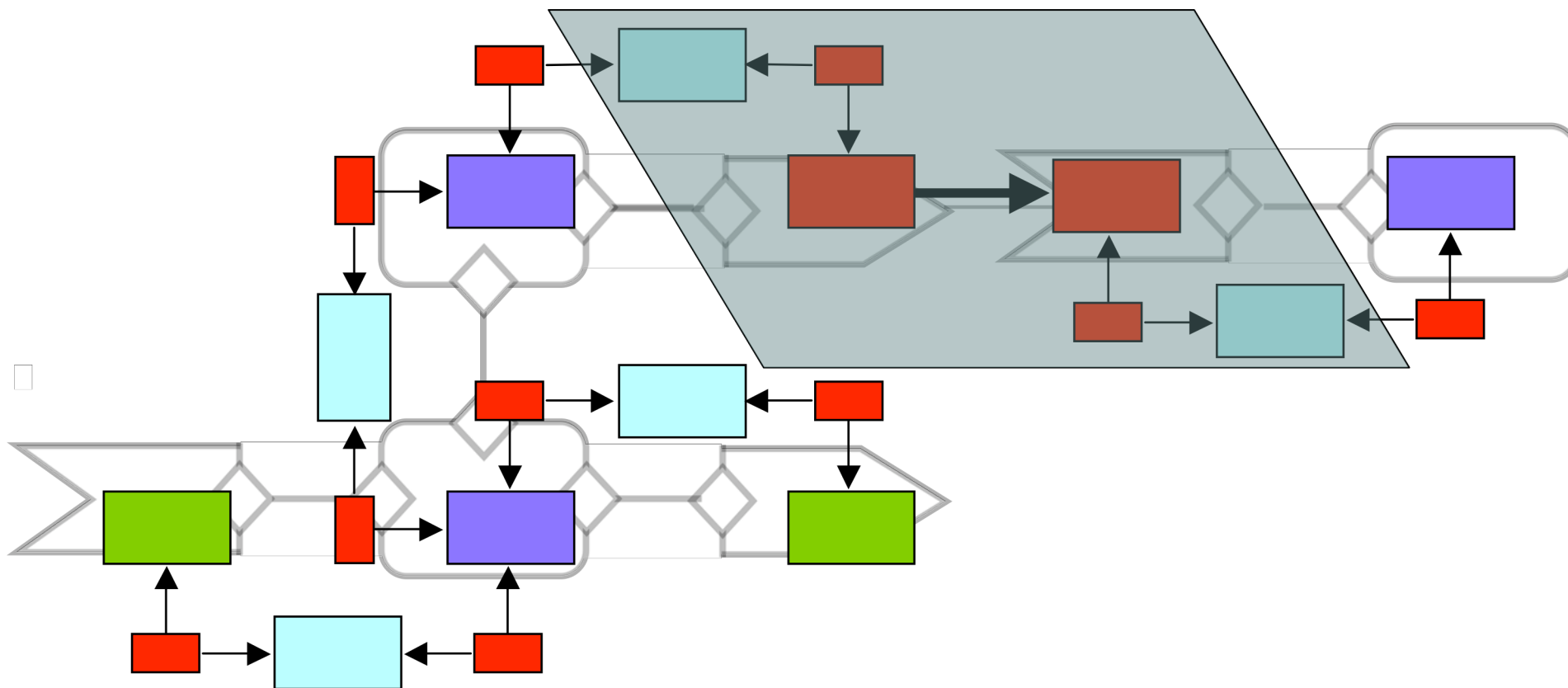


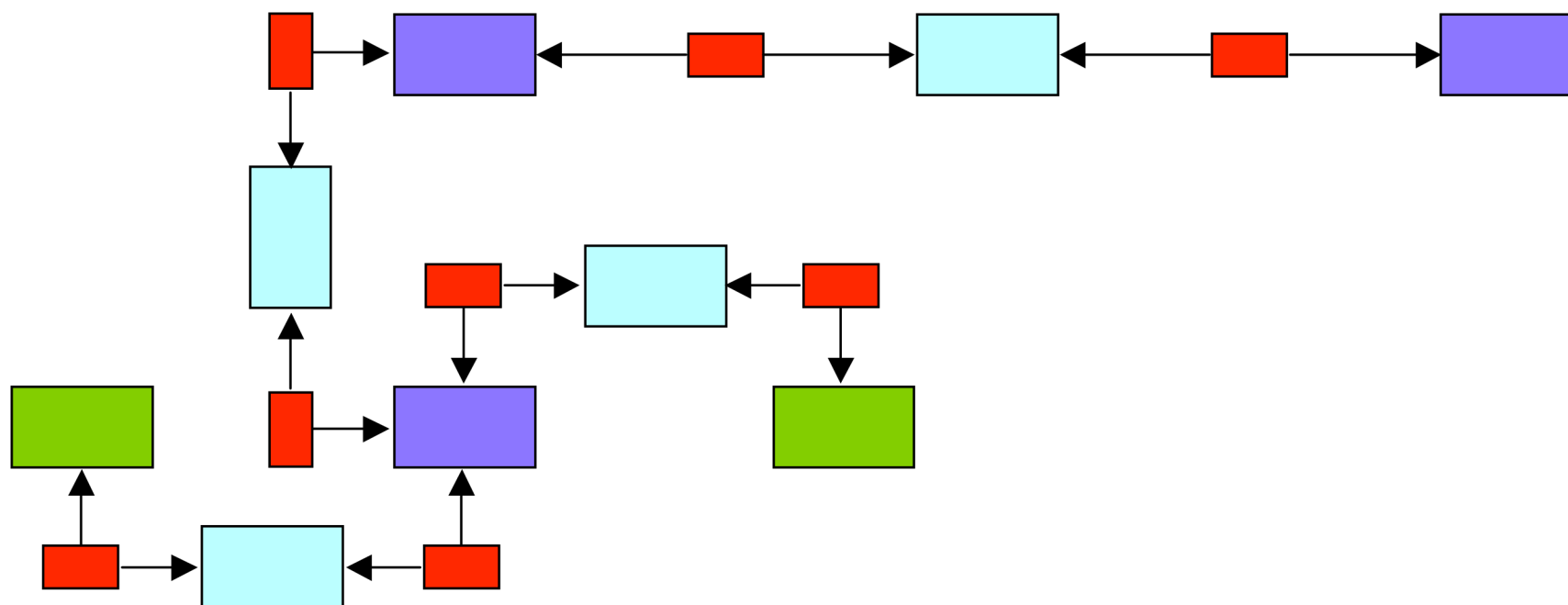






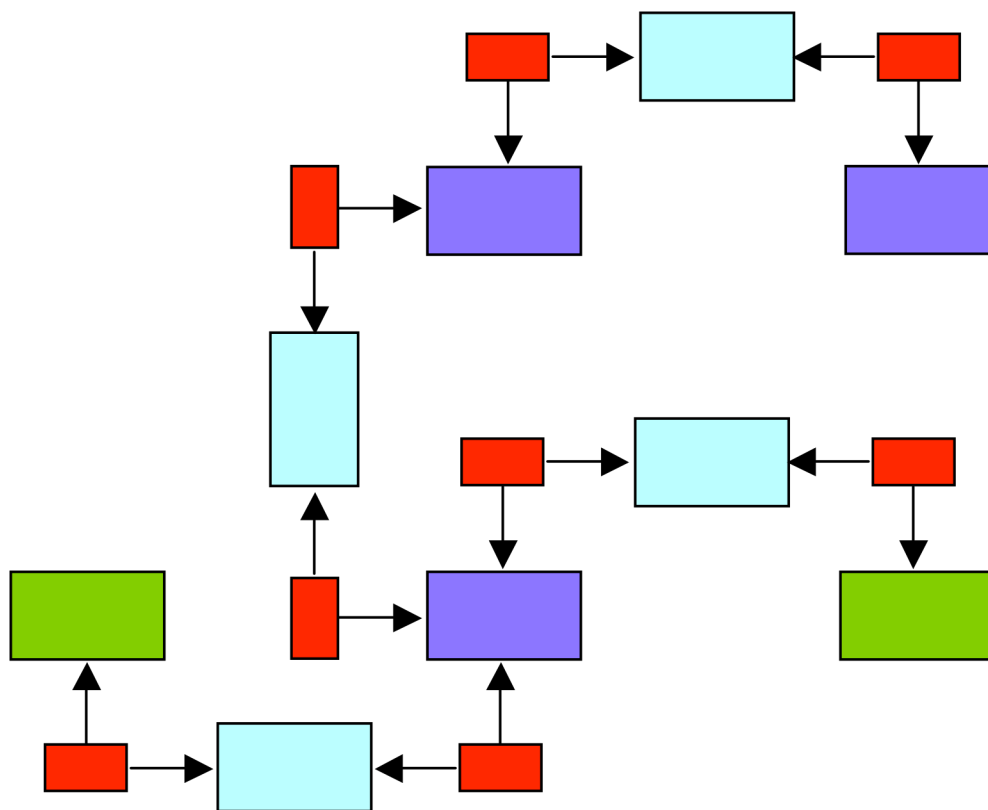






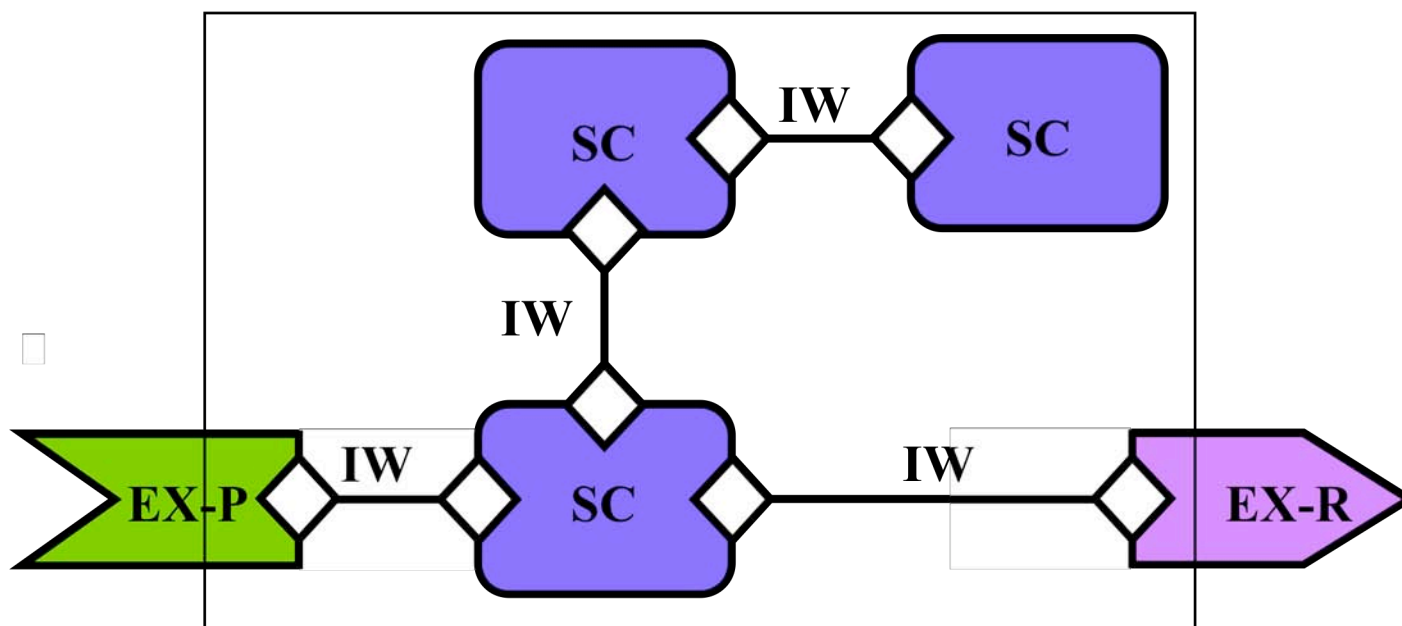
# Composition

16  
—  
18

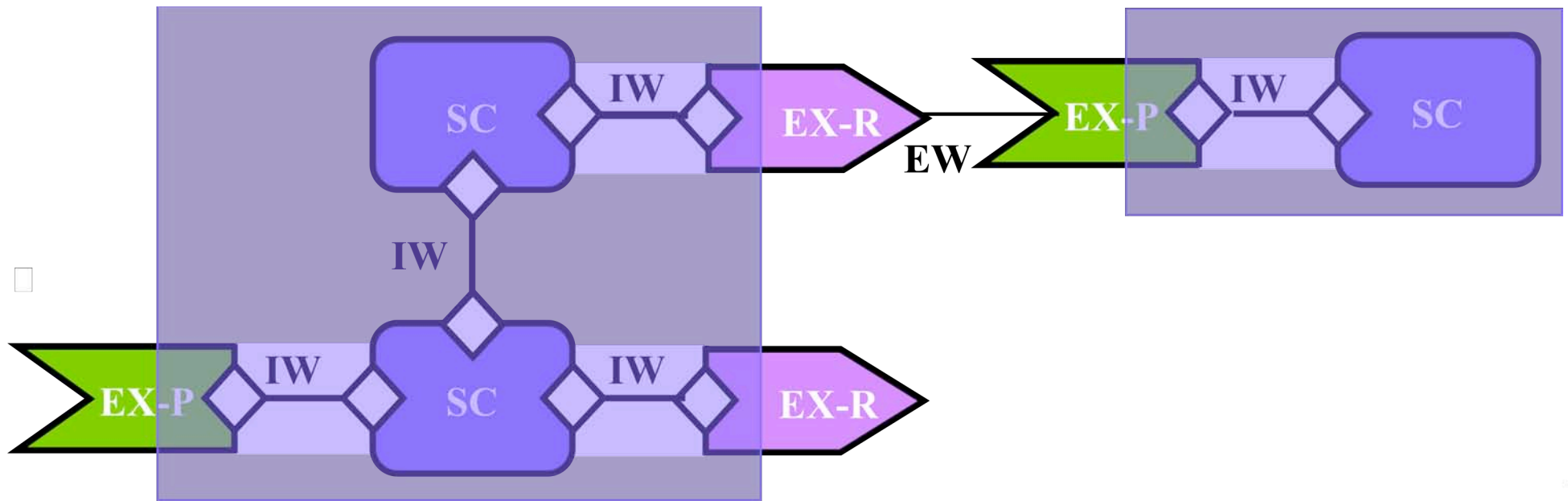


# Composition

16  
—  
18



# Composition



- Also available:
  - (Small) **Case Studies** (including Telecom with TILAB)
  - **SLA** attributes and statements
  - Deduction-based model of **discovery** using constraint-solving techniques for SLA
  - Configuration management, including **sessions** and **persistency**
  - Abstraction of business roles from **BPEL** components
- Planned research:
  - Operational semantics over core calculi
  - Analysis using UMC and PEPA
  - UML-notations
  - Model-transformation to SCA