An interface theory for service-oriented design

José Fiadeiro and Antónia Lopes

University of Leicester

UNIVERSIDADE DE LISBOA

Fiadeiro&Lopes@Aussois2011

Alfaro & Henzinger on CBD

Alfaro & Henzinger on CBD

- Component model/description

Alfaro & Henzinger on CBD

- Component model/description
  - answers the question “what does the component do?”

Alfaro & Henzinger on CBD

- Component model/description
  - answers the question "what does the component do?"
  - does not constrain the environment

Alfaro & Henzinger on CBD

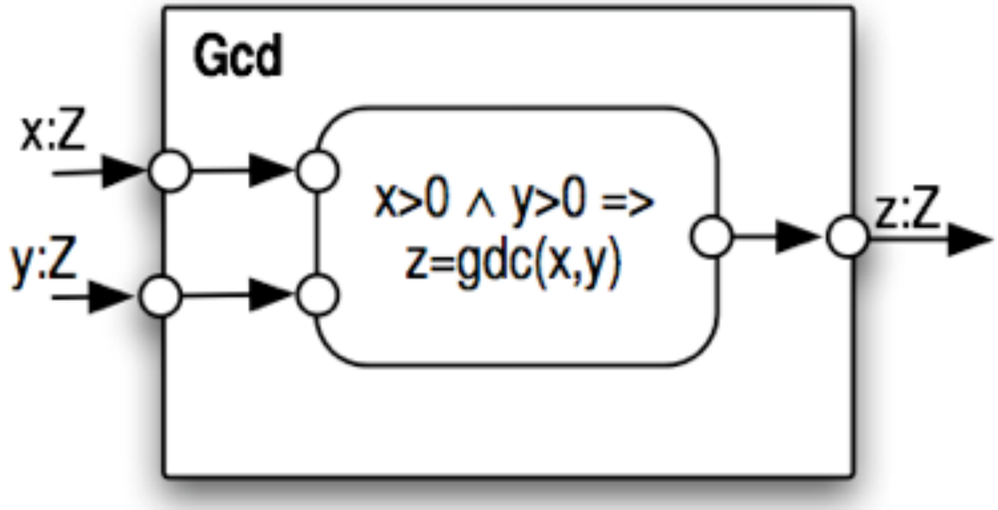
- Component model/description
  - answers the question "what does the component do?"
  - does not constrain the environment
  - examples: the body of a method (or a Pascal procedure), an I/O automaton, a Mealy machine, ...

Alfaro & Henzinger on CBD

- Component model/description
  - answers the question "what does the component do?"
  - does not constrain the environment
  - examples: the body of a method (or a Pascal procedure), an I/O automaton, a Mealy machine, ...
  - can be composed (subject to compatibility conditions)

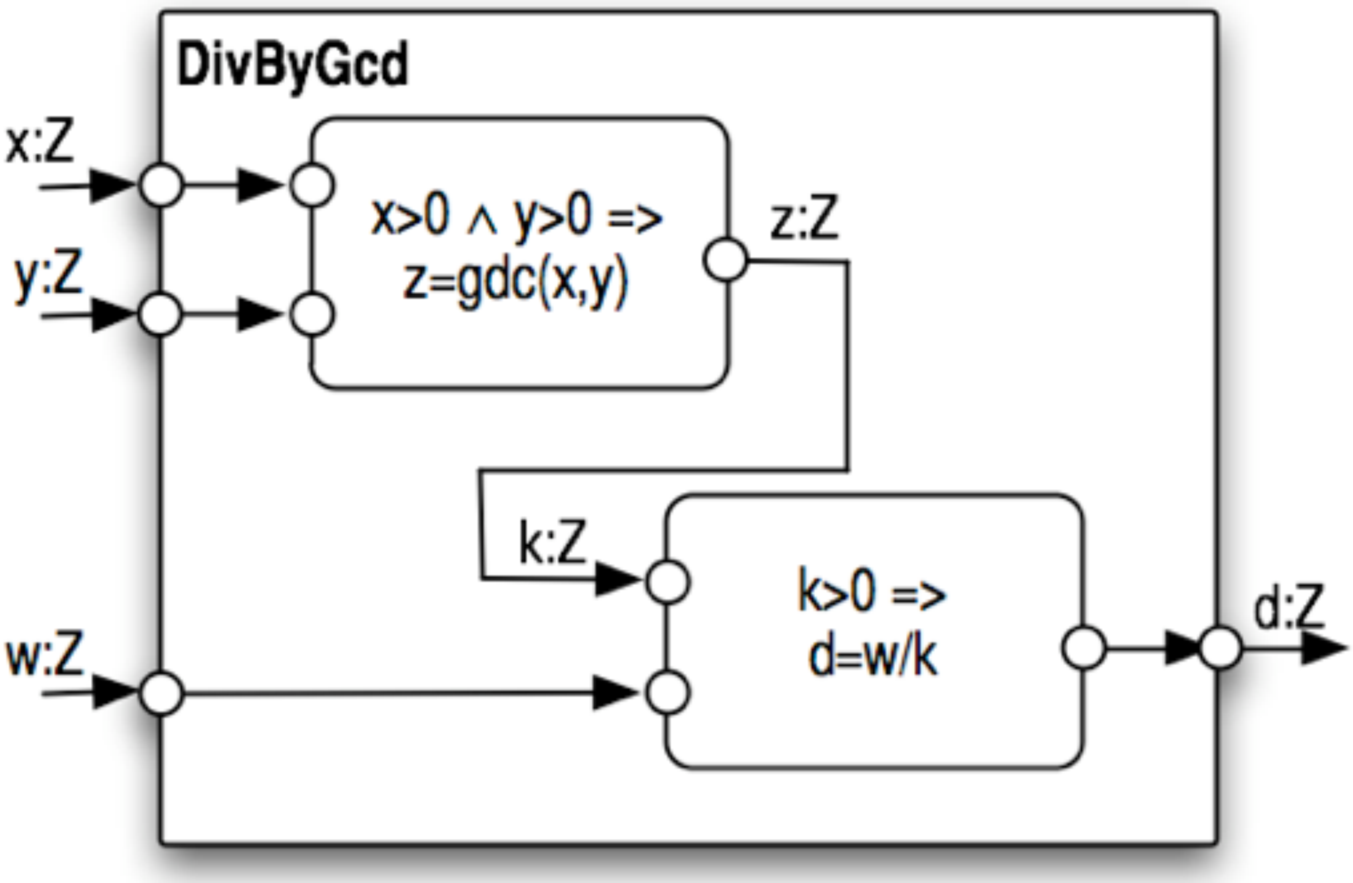
Alfaro & Henzinger on CBD

- Component model/description
  - relational nets:
    - a process consists of a set *I* of input ports, a set O of output ports and a satisfiable predicate on the set of ports
    - a channel is a pair of ports
    - the net is consistent in the sense that there is I/O valuation that satisfies the process predicates and the identities induced by the channels.



Alfaro & Henzinger on CBD

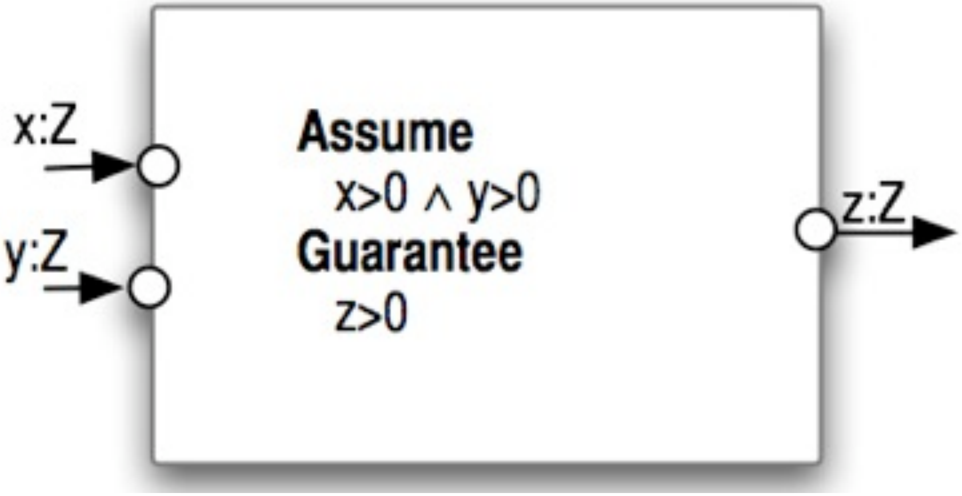
- Component model/description
  - relational nets:



O of output ports

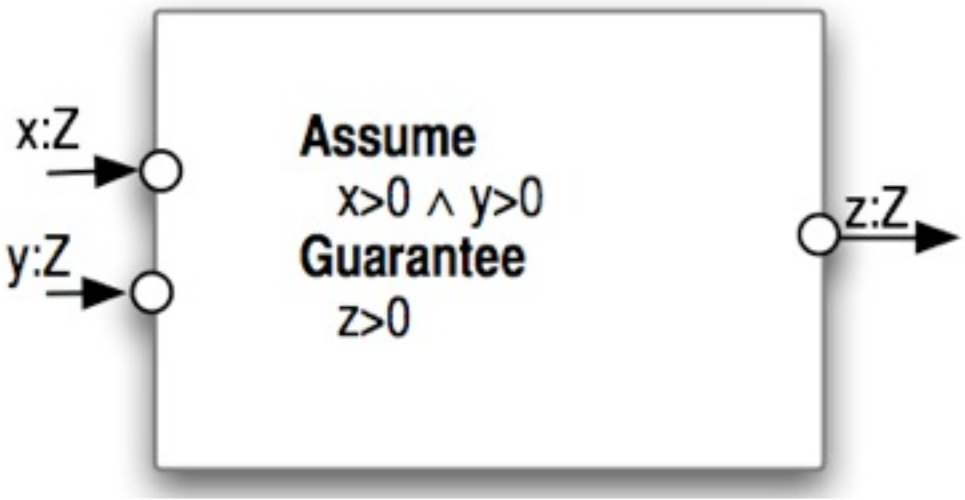
valuation that
induced by the

Alfaro & Henzinger on CBD



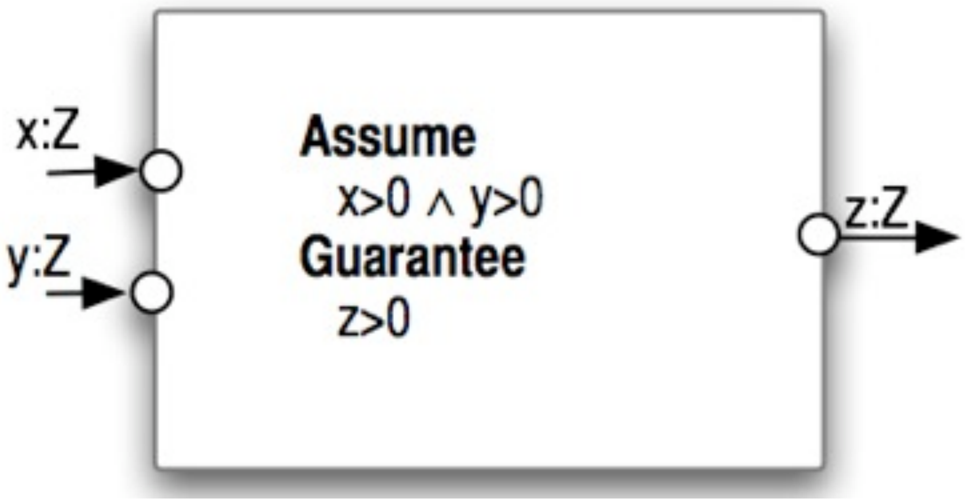
Alfaro & Henzinger on CBD

- Interface model/description



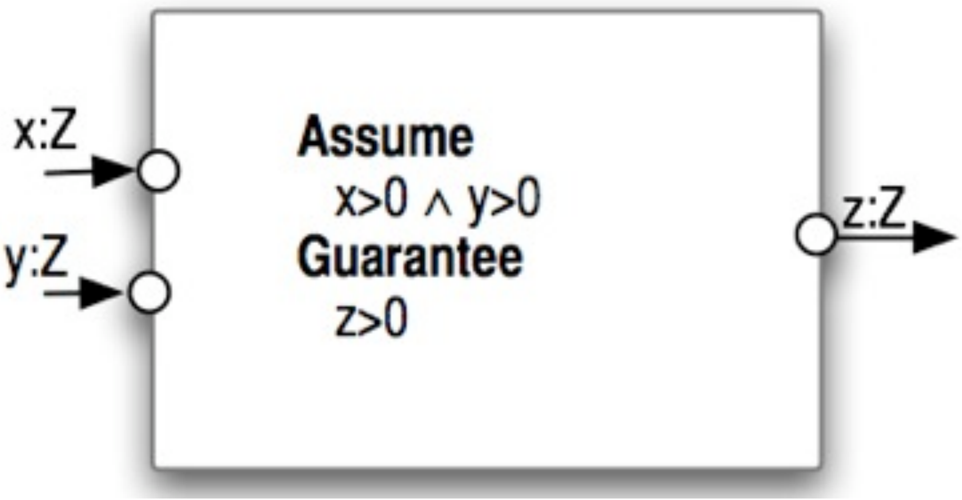
Alfaro & Henzinger on CBD

- Interface model/description
  - answers the question “how can the component be used?”



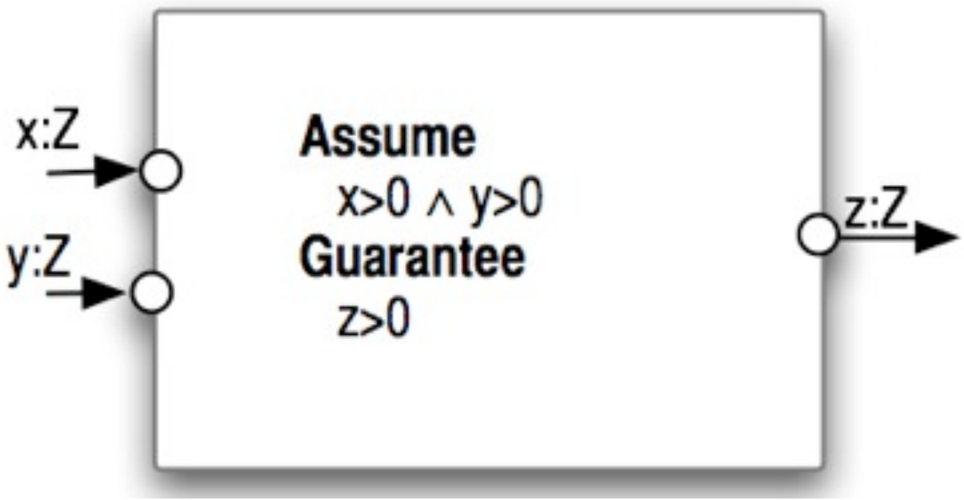
Alfaro & Henzinger on CBD

- Interface model/description
  - answers the question "how can the component be used?"
  - it constrains the environment by specifying the conditions under which the component expects to be used



Alfaro & Henzinger on CBD

- Interface model/description
  - answers the question "how can the component be used?"
  - it constrains the environment by specifying the conditions under which the component expects to be used
  - examples: parameter types, design by contract (assume/ guarantee conditions), interface automata, ...



Alfaro & Henzinger on CBD

Alfaro & Henzinger on CBD

- Interfaces vs components

Alfaro & Henzinger on CBD

- Interfaces vs components
  - interfaces support top-down design (through refinement) whereas components support bottom-up design (through abstraction)

Alfaro & Henzinger on CBD

- Interfaces vs components
  - interfaces support top-down design (through refinement) whereas components support bottom-up design (through abstraction)
  - they are related by a notion of implementation; interfaces are normally required to be implementable

Alfaro & Henzinger on CBD

- Interfaces vs components
  - interfaces support top-down design (through refinement) whereas components support bottom-up design (through abstraction)
  - they are related by a notion of implementation; interfaces are normally required to be implementable
  - ideally, implementation is compositional (which is the purpose of component-based design)

how is SOC different from CBD?

how is SOC different from CBD?

- two different notions of composition:
  - CBD is *integration-oriented* – "the idea of component-based development is to industrialise the software development process by producing software applications by assembling prefabricated software components" (A. Elfatatry. Dealing with change: components versus services. CACM, 50(8), 2007)

how is SOC different from CBD?

- two different notions of composition:
  - CBD is *integration-oriented* – "the idea of component-based development is to industrialise the software development process by producing software applications by assembling prefabricated software components" (A. Elfatatry. Dealing with change: components versus services. CACM, 50(8), 2007)
  - Hence, interfaces for CBD must describe the means through which software elements can be plugged together to build a product.

how is SOC different from CBD?

- two different notions of comp
  - CBD is *integration-orient*
    based development is t
    development process b
    applications by assemb
    components" (A. Elfatatry. D
    services. CACM, 50(8), 2007)
  - Hence, interfaces for CBD must describe the means through which software elements can be plugged together to build a product.

Interfaces such as assume/guarantee fall into this category: they specify the combinations of input values that components implementing an interface must accept (assumptions) and the combinations of output values that the environment can expect from them (guarantees).

Fiadeiro&Lopes@Aussois2011

how is SOC different from CBD?

- two different notions of composition:
  - CBD is *integration-oriented* – “the idea of component-based development is to industrialise the software development process by producing software applications by assembling prefabricated software components” (A. Elfatatry. Dealing with change: components versus services. CACM, 50(8), 2007)
  - Hence, interfaces for CBD must describe the means through which software elements can be plugged together to build a product.

how is SOC different from CBD?

how is SOC different from CBD?

- two different notions of composition:
  - SOC is *interaction-oriented* – services respond to the necessity for separating “need from the need- fulfilment mechanism” [Elfatatry] and address the ability of software elements to engage with other parties to pursue a given business goal.

how is SOC different from CBD?

- two different notions of composition:
  - SOC is *interaction-oriented* – services respond to the necessity for separating "need from the need- fulfilment mechanism" [Elfatatry] and
    software elements to en
    pursue a given busine

For example, we can design a seller that may need to use an external supplier if the local stock is low (the need); the discovery and selection of, and binding to, a specific supplier (the need-fulfilment mechanism) are not part of the design of the seller but performed, at run time, by the underlying middleware (SOA)

how is SOC different from CBD?

- two different notions of composition:
  - SOC is *interaction-oriented* – services respond to the necessity for separating "need from the need- fulfilment mechanism" [Elfatatry] and address the ability of software elements to engage with other parties to pursue a given business goal.
  - Hence, service interfaces must describe the properties that are provided (so that services can be discovered) as well as those that may be required from external services (so that the middleware can select a proper provider).

how is SOC different from CBD?

- two diff… …omposition:
  - …ces respond to the …the need- fulfilment …s the ability of so… …ith other parties to pursue a given business goal.

The latter are not assumptions on the environment as in CBD – in a sense, a service creates the environment that it needs to deliver what it promises.

  - Hence, service interfaces must describe the properties that are provided (so that services can be discovered) as well as those that may be required from external services (so that the middleware can select a proper provider).

Fiadeiro&Lopes@Aussois2011

how is SOC different from CBD?

how is SOC different from CBD?

- two different computational models:
  - CBD is *I/O-oriented* – typical component algebras are synchronous: the client knows and invokes the server with input values and waits for the return.

how is SOC different from CBD?

- two different computational models:
  - CBD is *I/O-oriented* – typical component algebras are synchronous: the client knows and invokes the server with input values and waits for the return.
  - SOC is intrinsically *asynchronous* and *conversational*.

how is SOC different from CBD?

- two different computational models:
  - CBD is *I/O-oriented* – typical component algebras are synchronous: the client knows and invokes the server with input values and waits for the return.
  - SOC is intrinsically *asynchronous* and *conversational*.
    - However, most existing models for choreography are indeed synchronous...

how is SOC different from CBD?

- two different computational models:
  - CBD is *I/O-oriented* – typical component algebras are synchronous: the client knows and invokes the server with input values and waits for the return.
  - SOC is intrinsically *asynchronous* and *conversational*.
    - However, most existing models for choreography are indeed synchronous...
    - Our approach is orchestration-oriented: we propose to model the workflow through which a service is orchestrated as being executed by a network of processes that interact asynchronously and offer interaction-points to which clients and external services (executed by their own networks) can bind.

our research questions

our research questions

- What is a suitable notion of interface for such asynchronous networks of processes that deliver a service?

our research questions

- What is a suitable notion of interface for such asynchronous networks of processes that deliver a service?
  - What is an asynchronous network of processes?

our research questions

- What is a suitable notion of interface for such asynchronous networks of processes that deliver a service?
  - What is an asynchronous network of processes?
- What notion of interface composition is suitable for the loose coupling of the business processes that orchestrate the interfaces?

asynchronous relational nets (ARNs)

Fiadeiro&Lopes@Aussois2011

asynchronous relational nets (ARNs)

- Highlights

asynchronous relational nets (ARNs)

- Highlights
  - Services are delivered by networks of processes/ components – as in SCA (the Service Component Architecture)

asynchronous relational nets (ARNs)

- Highlights
  - Services are delivered by networks of processes/ components – as in SCA (the Service Component Architecture)
  - Processes interact by exchanging messages

asynchronous relational nets (ARNs)

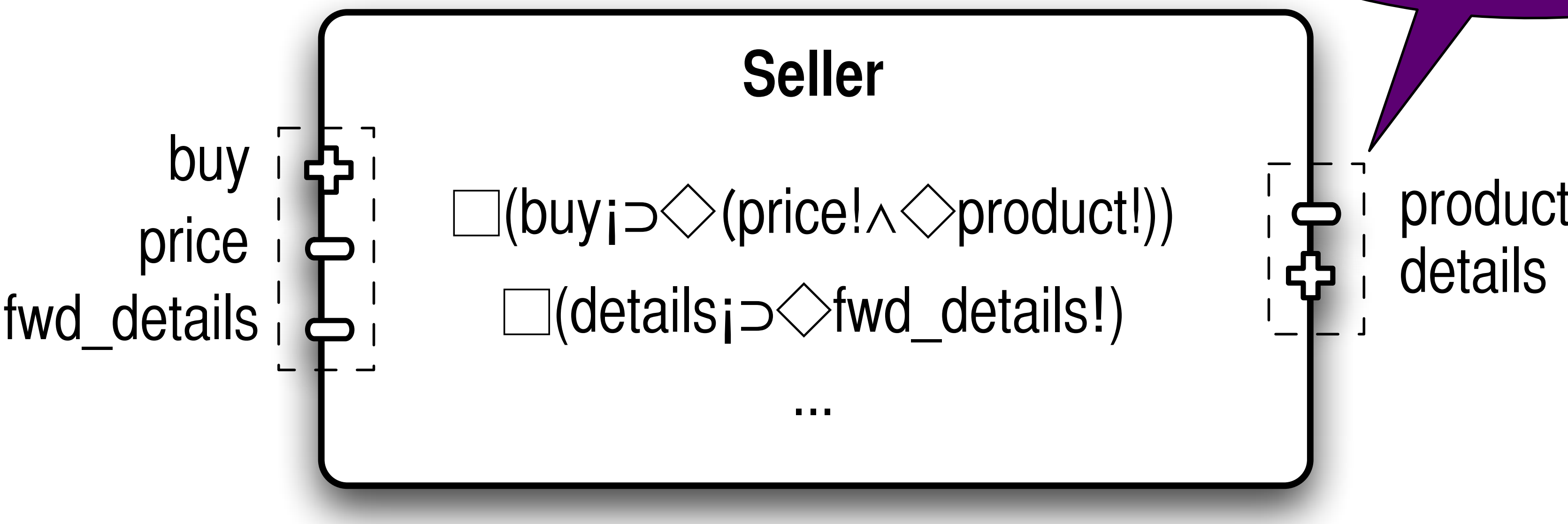
- Highlights
  - Services are delivered by networks of processes/components – as in SCA (the Service Component Architecture)
  - Processes interact by exchanging messages
  - Messages are transmitted through channels

asynchronous relational nets (ARNs)

- Highlights
  - Services are delivered by networks of processes/components – as in SCA (the Service Component Architecture)
  - Processes interact by exchanging messages
  - Messages are transmitted through channels
  - Temporal logic is used for describing processes and channels – actions consist of message delivery (m¡), processing (m?), discarding (m¿) or sending (m!)

processes

- A process consists of
  - A finite set of mutually-disjoint ports
  - A consistent set of LTL formulas



channels

- A channel consists of
  - A set of messages
  - A consistent set of LTL formulas over delivery and sending of messages

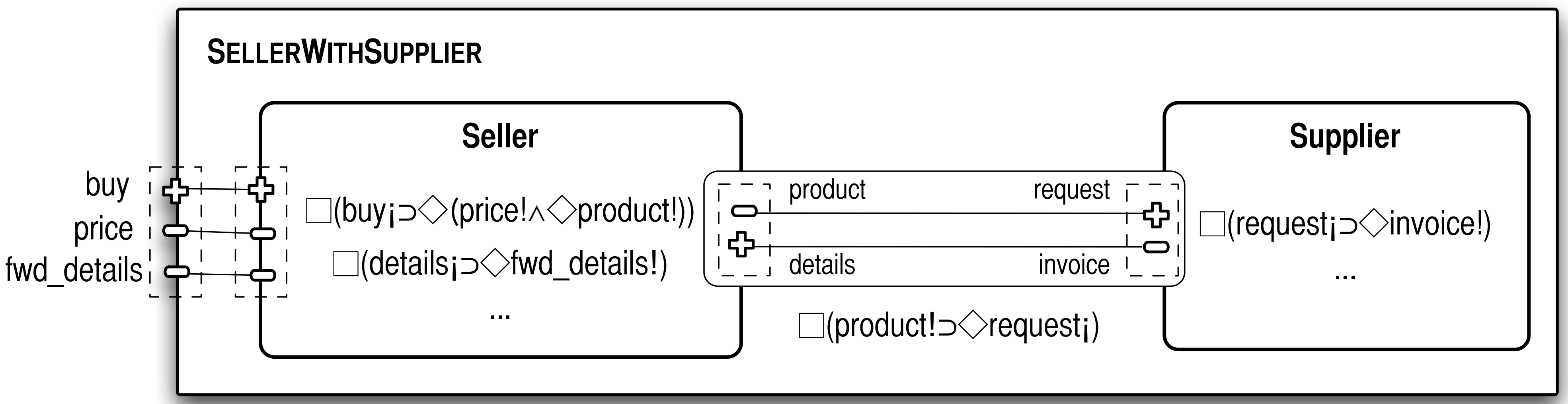
– e.g., $\Box(m! \supset \Diamond m¡)$

(the channel is reliable – it delivers the message once it is published)

ARNs

ARNs

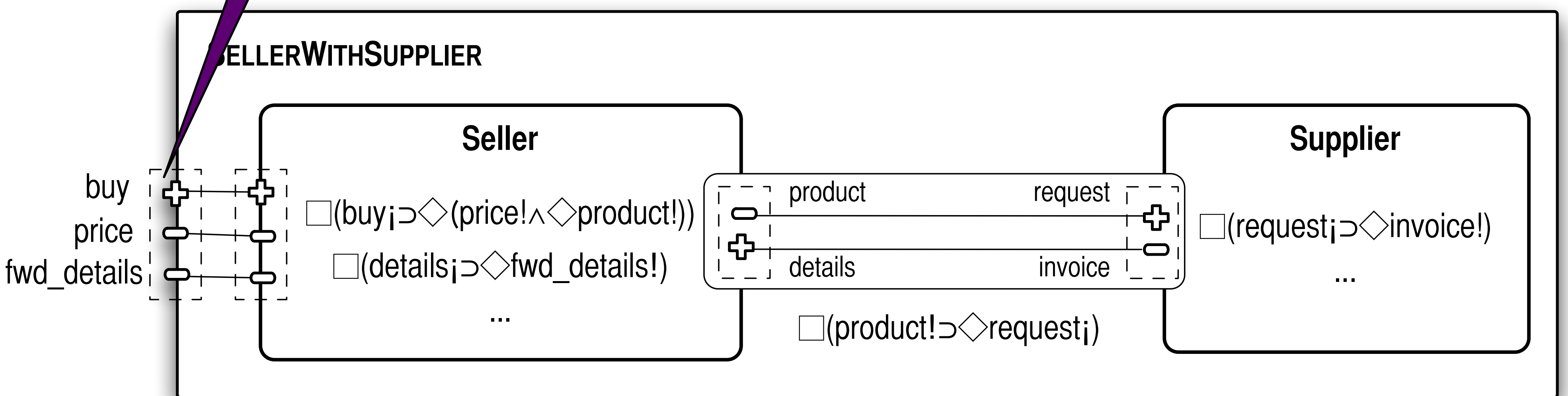
- An ARN is a simple graph where
  - Nodes are labelled with processes
  - Edges are labelled with connections (wires+attachments)



ARNs

An interaction point: a port that is not connected

- ... e graph where
  - ... with processes
  - ... elled with connections (wires+attachments)
- ARN... can be composed by interconnecting inte... action-points (via channels)



Fiadeiro&Lopes@Aussois2011

consistency?

- Relational nets (de Alfaro & Henzinger) are required to be jointly consistent – inputs and outputs match, i.e. the processes can communicate
- What about ARNs?
  - The set of infinite traces of an ARN α that are projected to models of all processes and channels is

$$\Lambda_\alpha = \{\lambda \in 2^{A_\alpha}{}^\omega : \forall p \in P(\lambda|_{A_p} \in \Lambda_{\Phi_p}) \wedge \forall c \in C(\lambda|_{A_c} \in \Lambda_{\Phi_c})\}$$

  - Consistency means $\Lambda_\alpha \neq \varnothing$

Fiadeiro&Lopes@Aussois2011

progress-enabled ARNs

- We consider instead finite behaviours
  - The set of finite traces that are projected to prefixes of models of all processes and channels is

$$\Pi_\alpha = \{\pi \in {2^{A_\alpha}}^* : \forall p \in P(\pi|_{A_p} \in \Pi_{\Phi_p}) \wedge \forall c \in C(\pi|_{A_c} \in \Pi_{\Phi_c})\}$$

  - An ARN α is progress enabled iff its processes are always able to make progress while interacting through the channels

$$\forall \pi \in \Pi_\alpha \exists A \subseteq A_\alpha (\pi \cdot A) \in \Pi_\alpha$$

the first tricky question...

- (When) is the composition of two progress-enabled ARNs progress enabled?
  - This needs to be understood in terms of a computational and communication model in which it is clear what dependencies exist between the different parties.
  - We take it to be the responsibility of processes to publish and process messages, and of channels to deliver them. This requires that processes are able to buffer incoming messages and that channels are able to buffer published messages, thus making them ‘co-operative’.

co-operative processes

- An ARN α is delivery-enabled in relation to an interaction point <p,M> iff
  - For every $(\pi.A) \in \prod_\alpha$ and $B \subseteq D_{<p,M>} = \{p.m_{\text{¡}} : m \in M\}$, $(\pi.B \cup (A \setminus D_{<p,M>}) \in \prod_\alpha$
  - That is, any prefix can be extended with any set of messages delivered at that interaction-point.

Fiadeiro&Lopes@Aussois2011

co-operative channels

- A channel $h=<M,\Phi>$ is publication-enabled iff
  - For every $(\pi.A) \in \prod_{\Phi}$ and $B \subseteq E_h=\{p.m!: m \in M\}$, $(\pi.B \cup (A \setminus E_h)) \in \prod_{\Phi}$
  - That is, any prefix can be extended by the publication of a set of messages, i.e. the channel should not prevent processes from publishing messages.

First theorem

- Let α be a composition of α_1 and α_2 by interconnecting interaction points <p_1,M_1> and <p_2,M_2> via a channel h.

  Then, α is progress-enabled if:

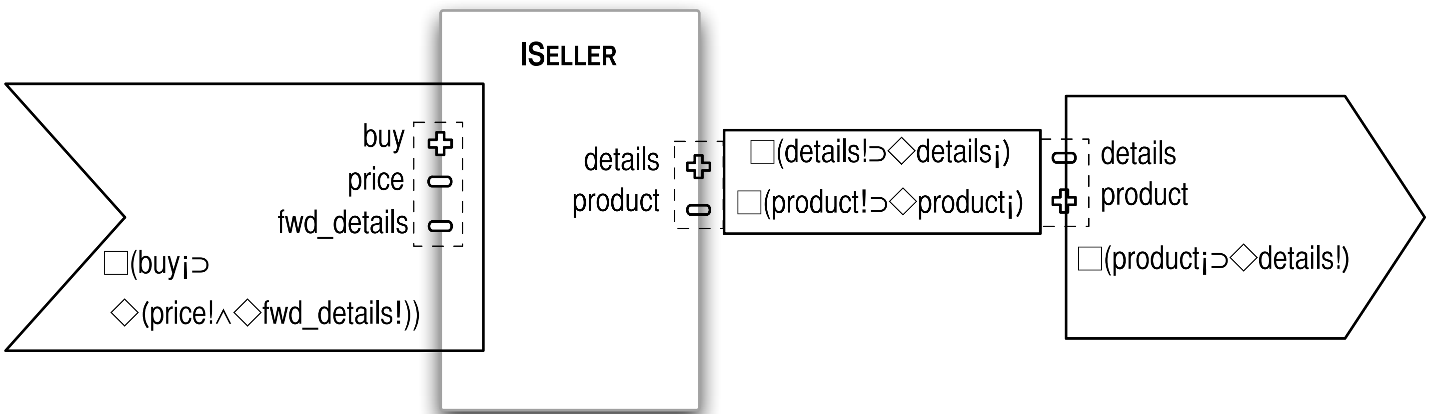
  - α_1 and α_2 are progress-enabled
  - α_1 and α_2 are delivery-enabled in relation to <p_1,M_1> and <p_2,M_2>, respectively
  - h is publication-enabled

Interfaces

- A service interface consists of:
  - Sets $I^{\rightarrow}$ (of provides-points) and $I^{\leftarrow}$ (of requires-points)
  - For every provides-point r, a process $<\{M_r\},\Phi_r>$
  - For every requires-point r:
    - a process $<\{M_r\},\Phi_r>$ that is delivery-enabled
    - a channel $<M_r,\Psi_r>$ that is progress-enabled

Interfaces

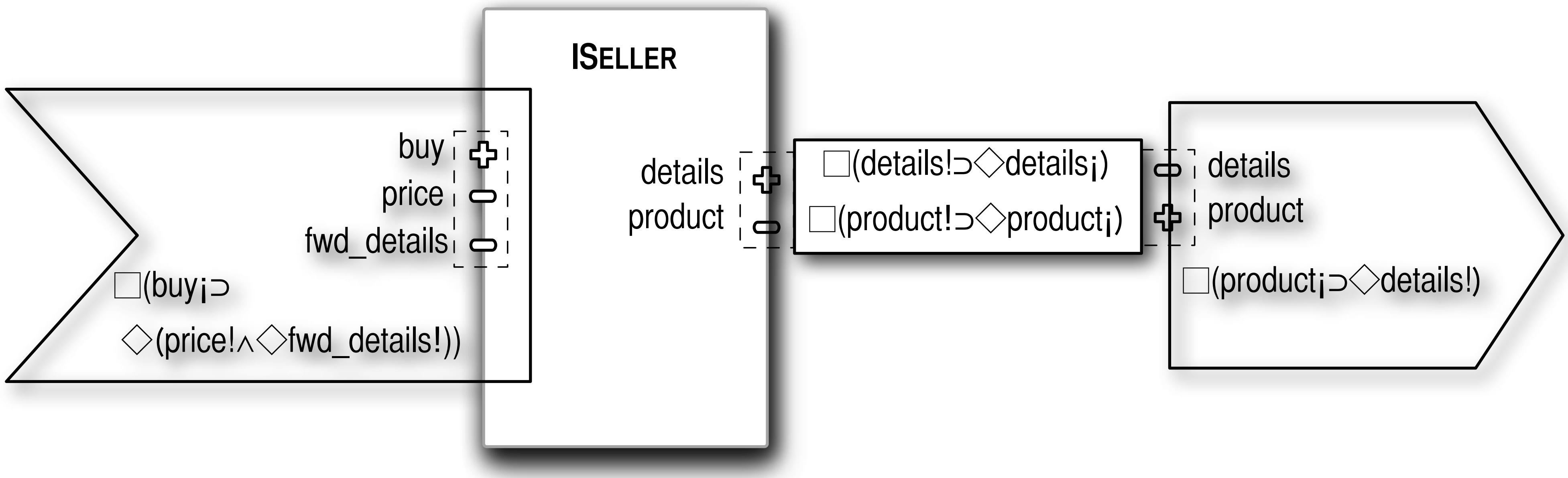
- A service interface consists of:
  - Sets $I^{\rightarrow}$ (of provides-points) and $I^{\leftarrow}$ (of requires-points)
  - For every provides-point r, a process $<\{M_r\},\Phi_r>$
  - For every requires-point r:
    - a process $<\{M_r\},\Phi_r>$ that is delivery-enabled
    - a channel $<M_r,\Psi_r>$ that is progress-enabled



Fiadeiro&Lopes@Aussois2011

Interfaces

- A service interface consists of:
  - Sets $I^{\rightarrow}$ (of provides-points) and $I^{\leftarrow}$ (of requires-points)
  - For every provides-point r, a process $<\{M_r\},\Phi_r>$
  - For every requires-point r:
    - a process $<\{M_r\},\Phi_r>$ that is delivery-enabled
    - a channel $<M_r,\Psi_r>$ that is progress-enabled



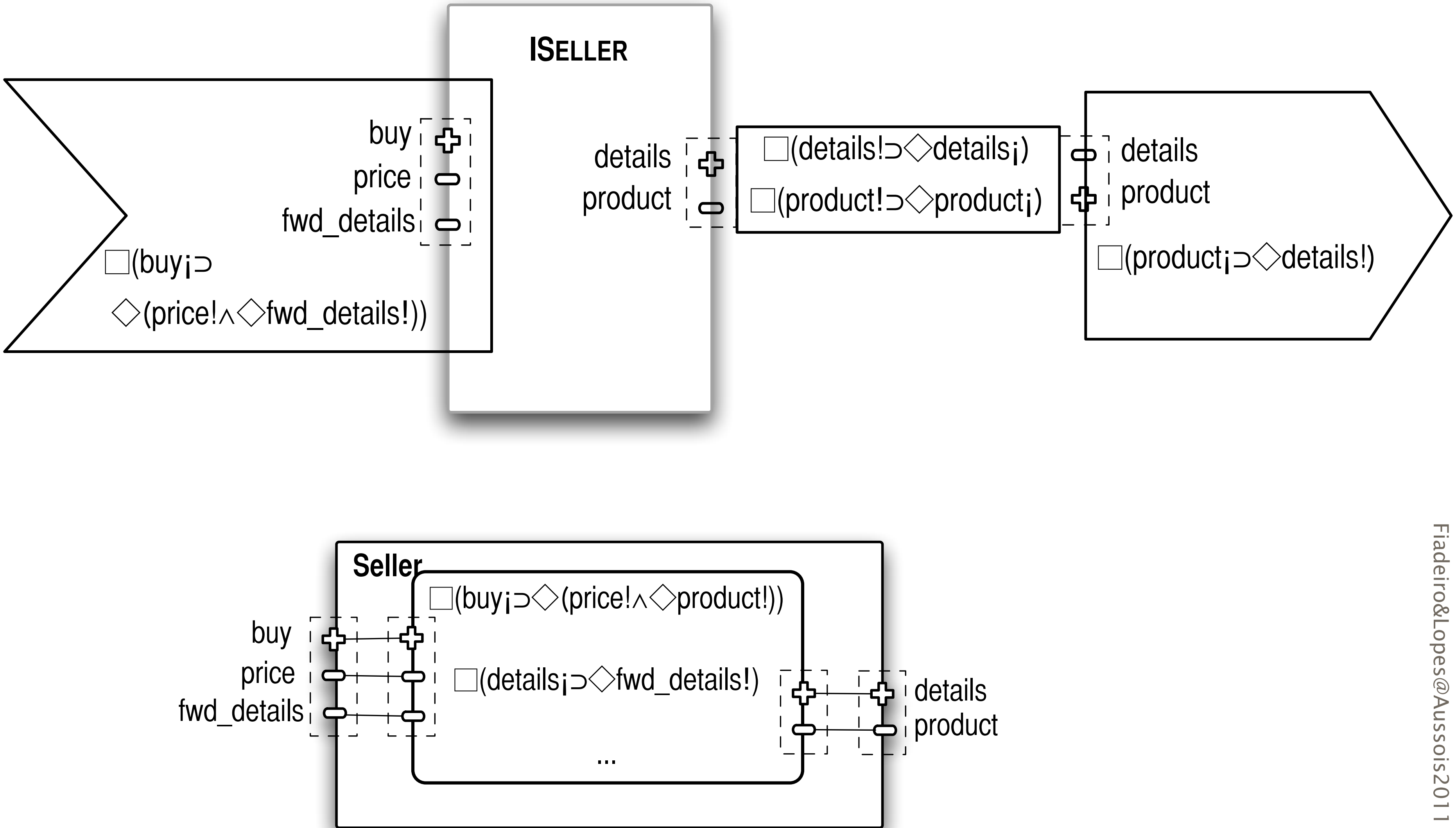
Fiadeiro&Lopes@Aussois2011

Orchestration

- An orchestration of a service interface consists of:
  - An ARN α that is progress-enabled and delivery-enabled in relation to all its interaction points
  - A 1–1 correspondence between the interaction points of the ARN and the interface points

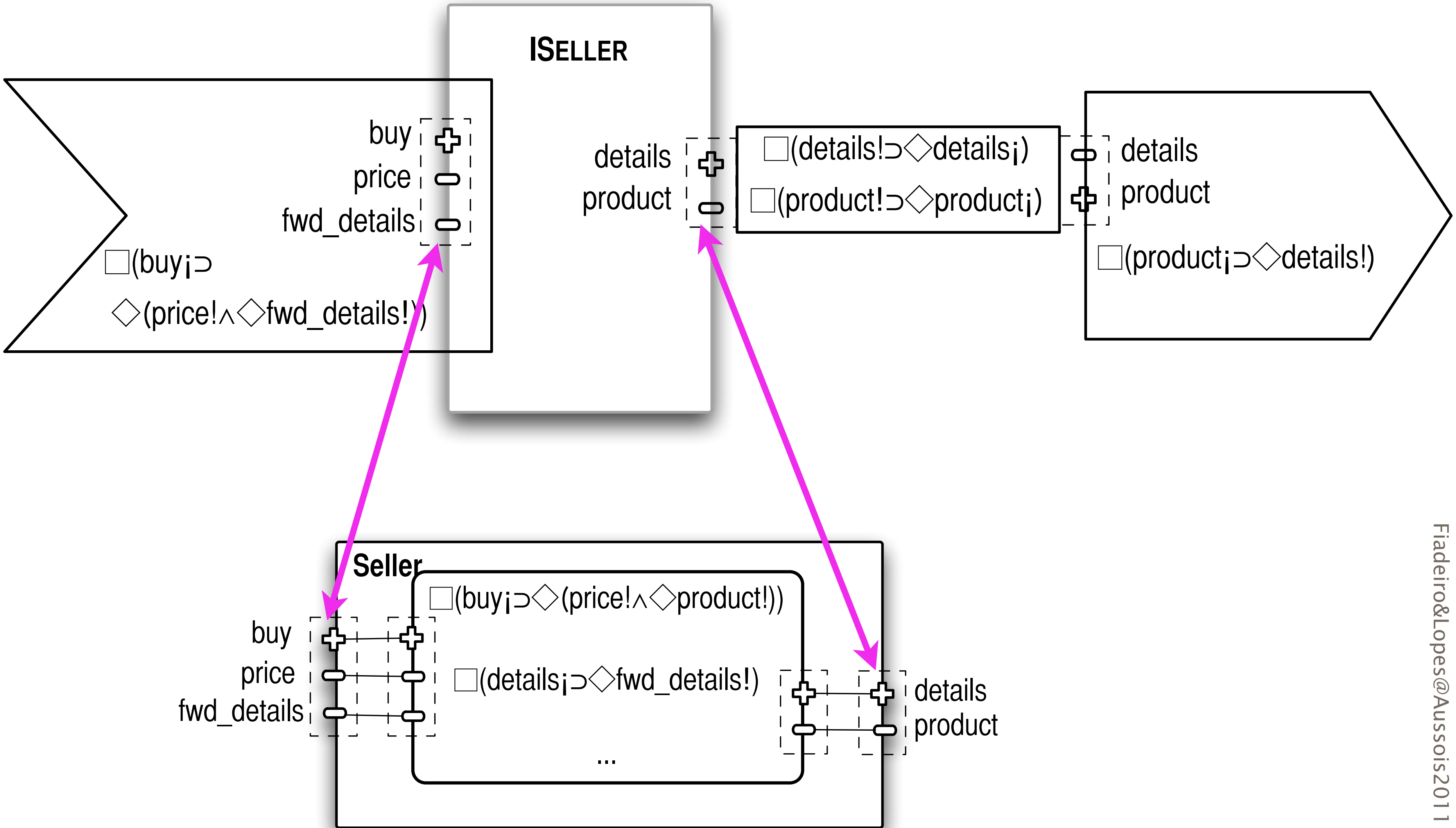
  such that all the properties of the provides-points are entailed by the ARN that consists of the composition of α with the requires-points and associated channels.

Orchestration

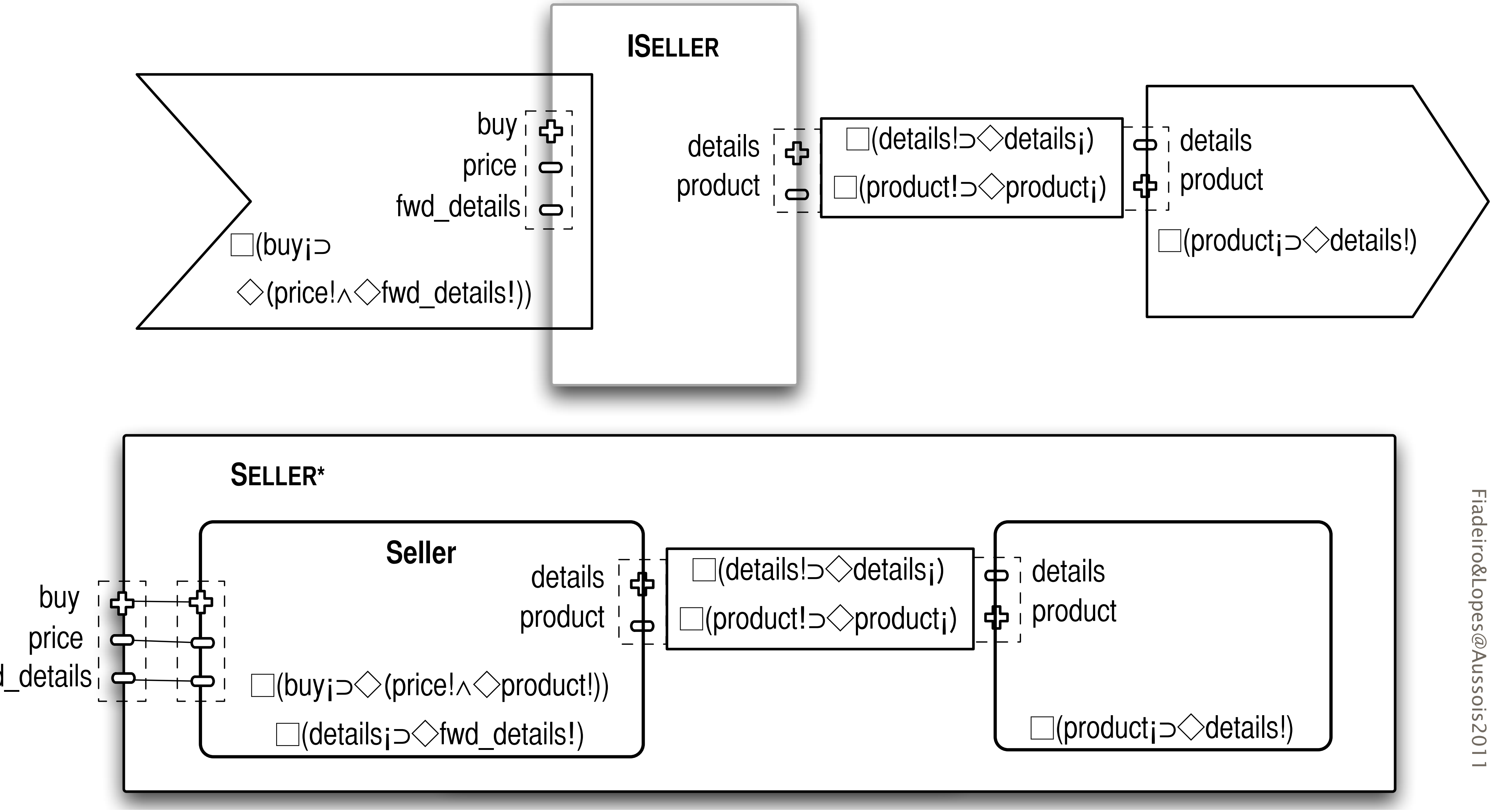


Fiadeiro&Lopes@Aussois2011

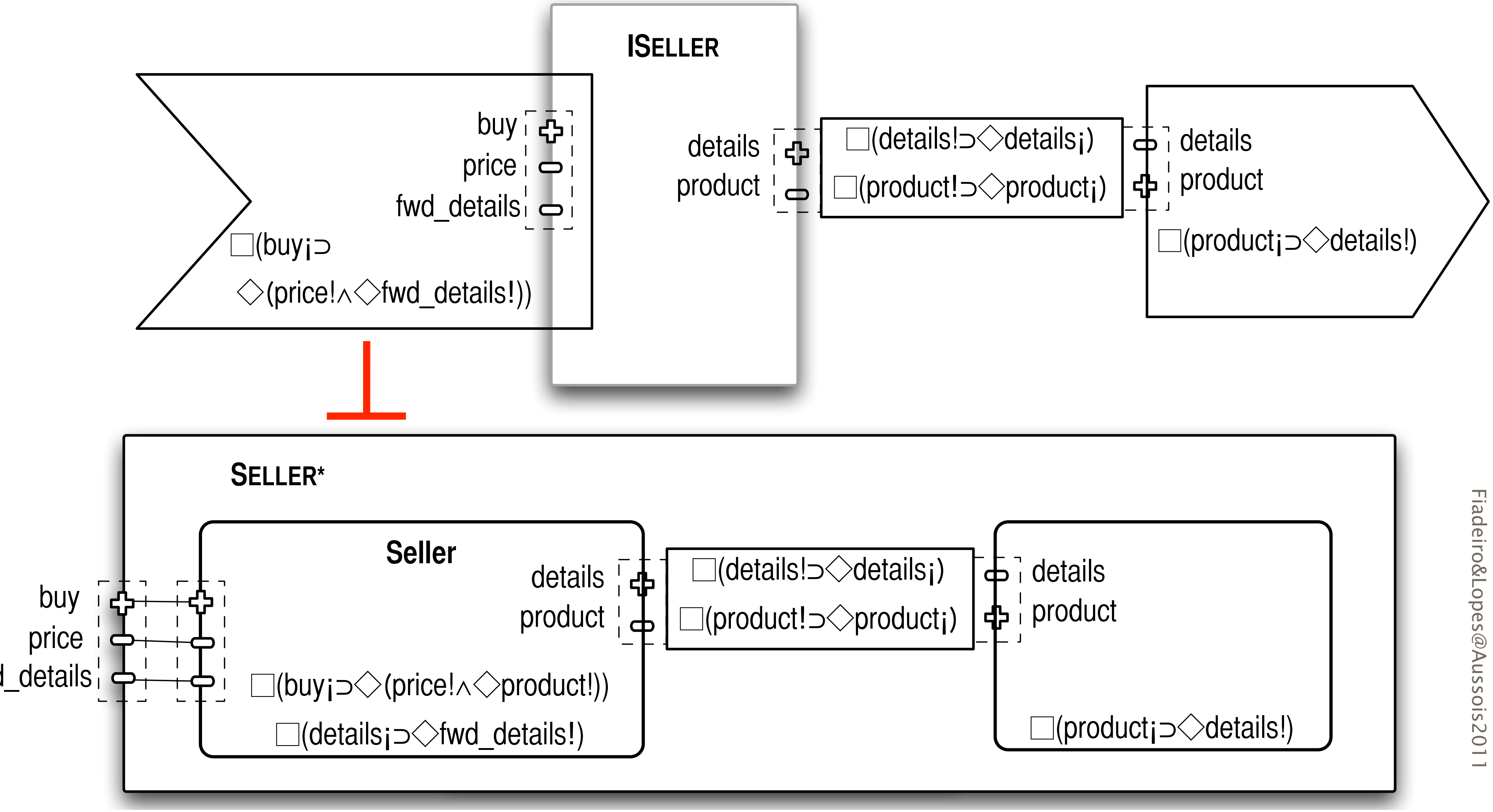
Orchestration



Orchestration

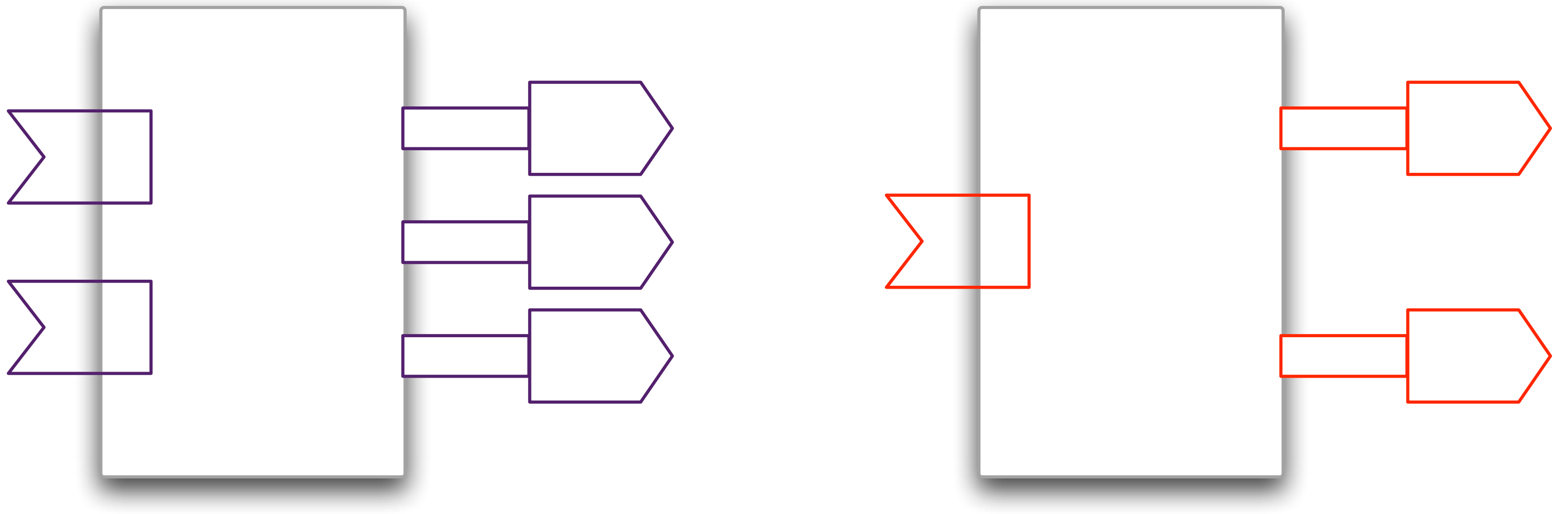


Orchestration

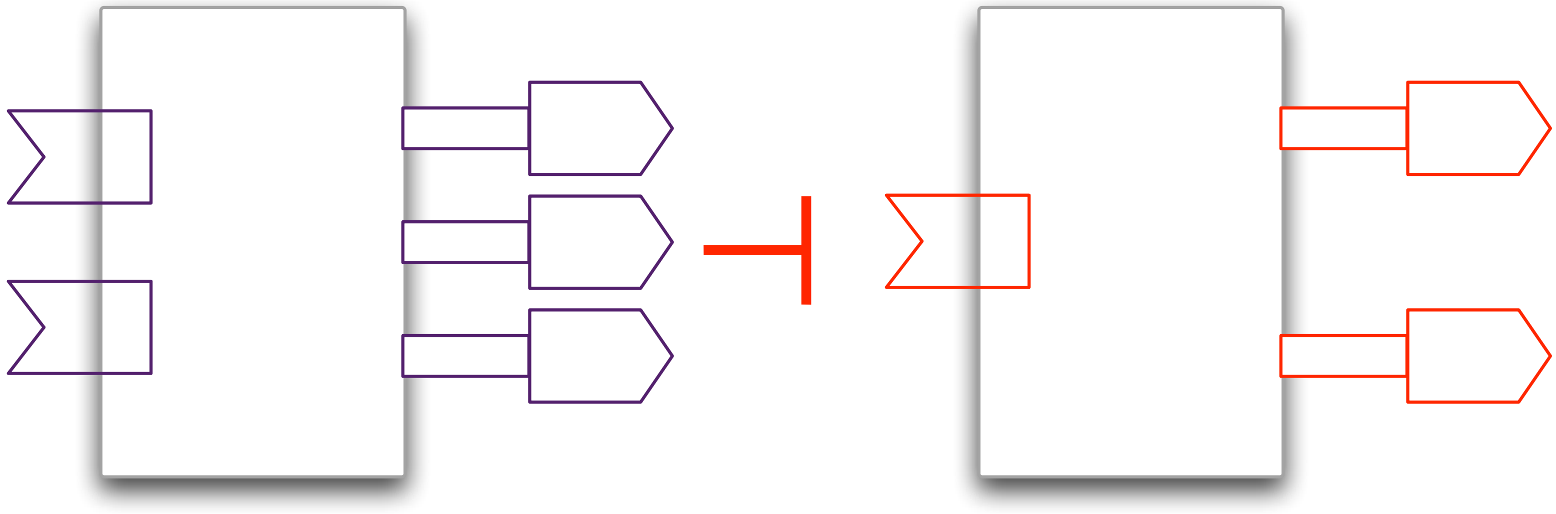


Fiadeiro&Lopes@Aussois2011

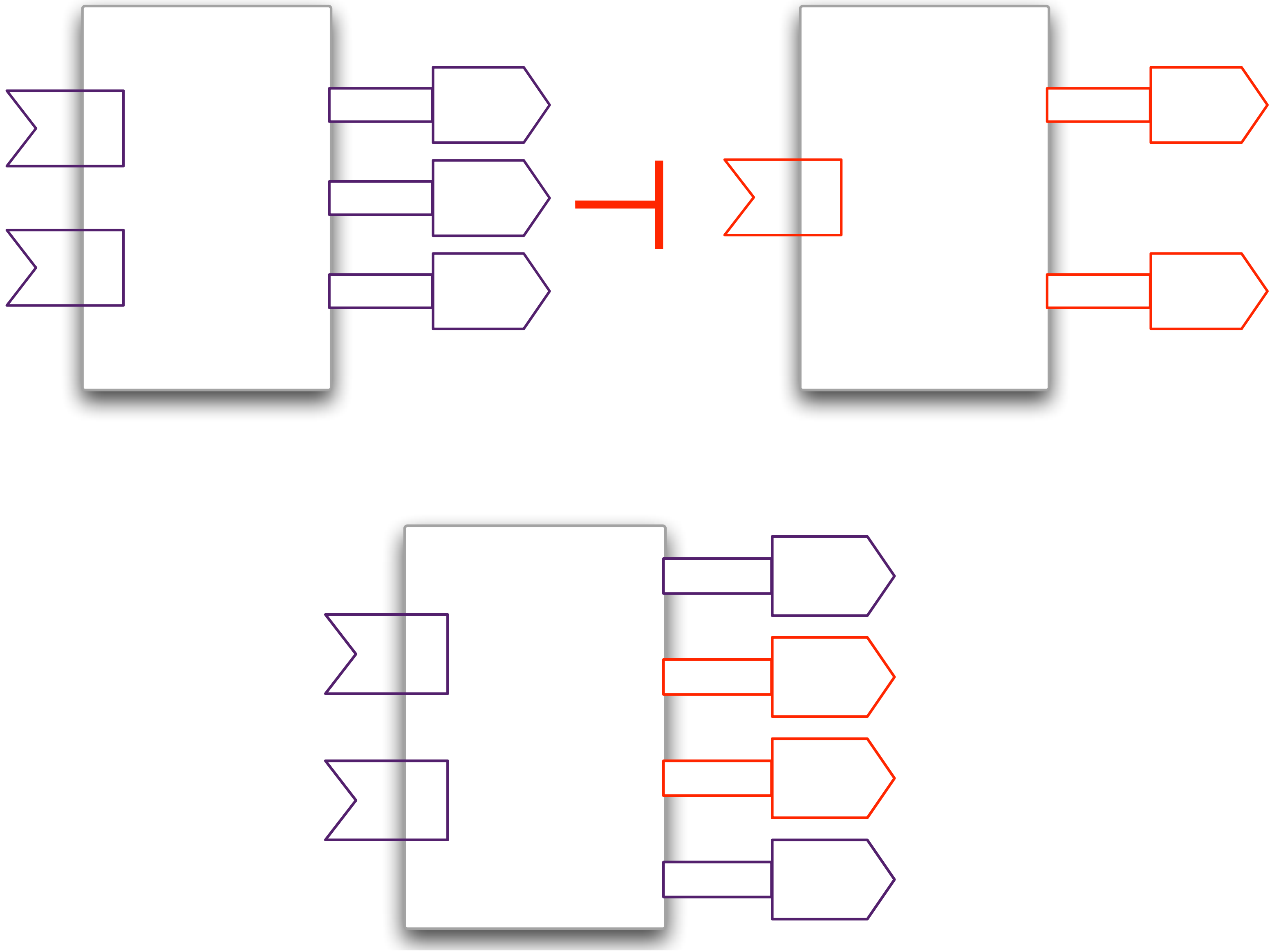
Match and compose



Match and compose



Match and compose



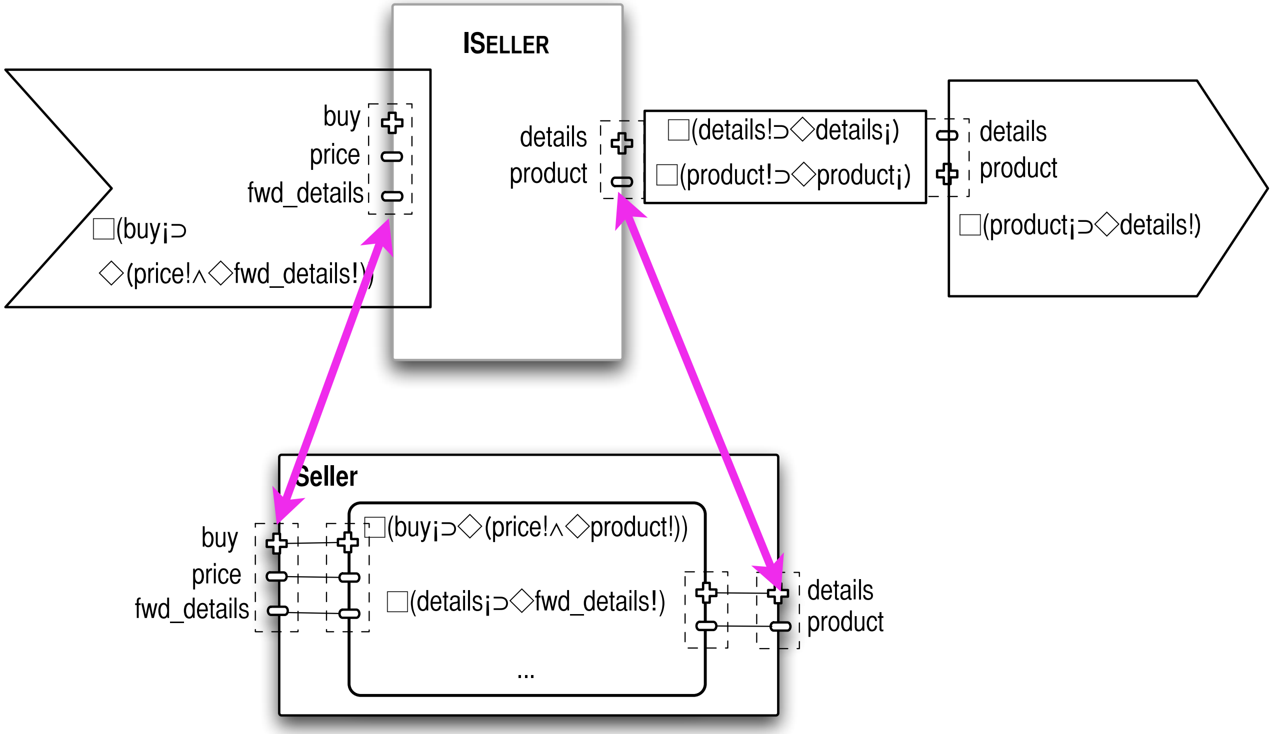
Fiadeiro&Lopes@Aussois2011

Second theorem – compositionality

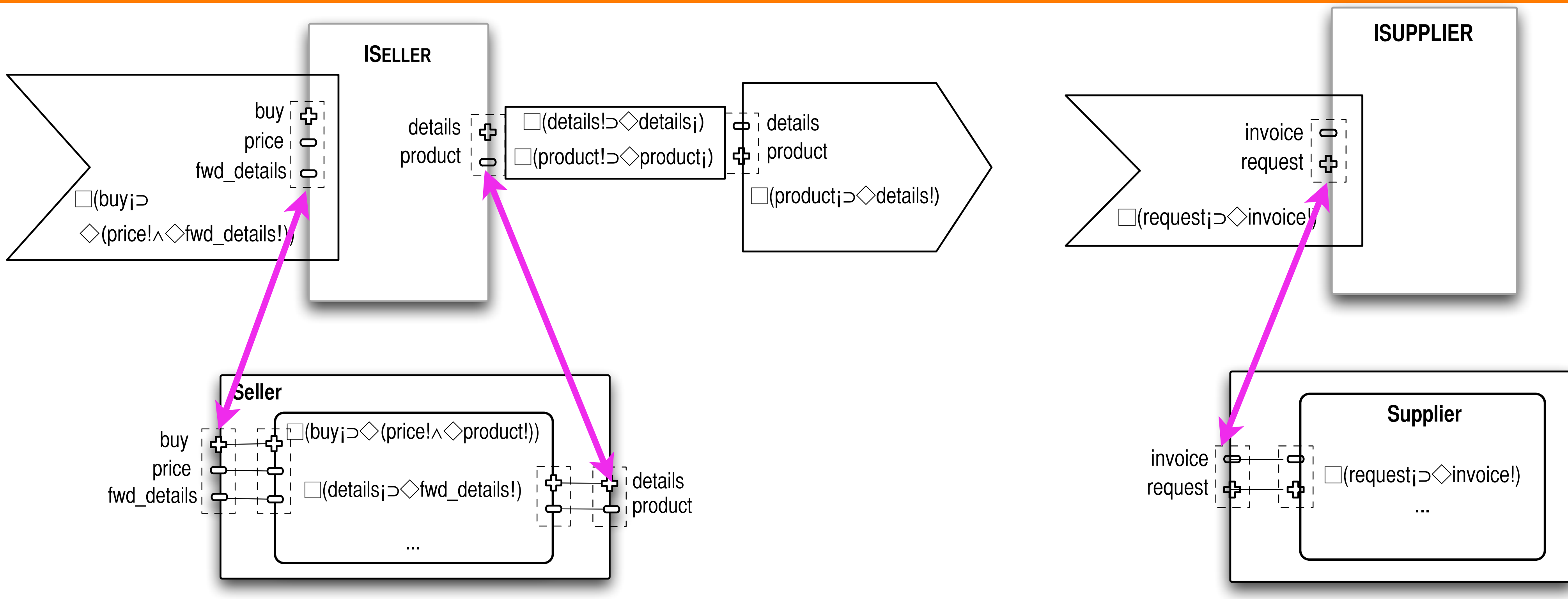
- The composition of the orchestrations of compatible interfaces is an orchestration of the composition of the interfaces.

Second theorem – compositionality

Second theorem – compositionality

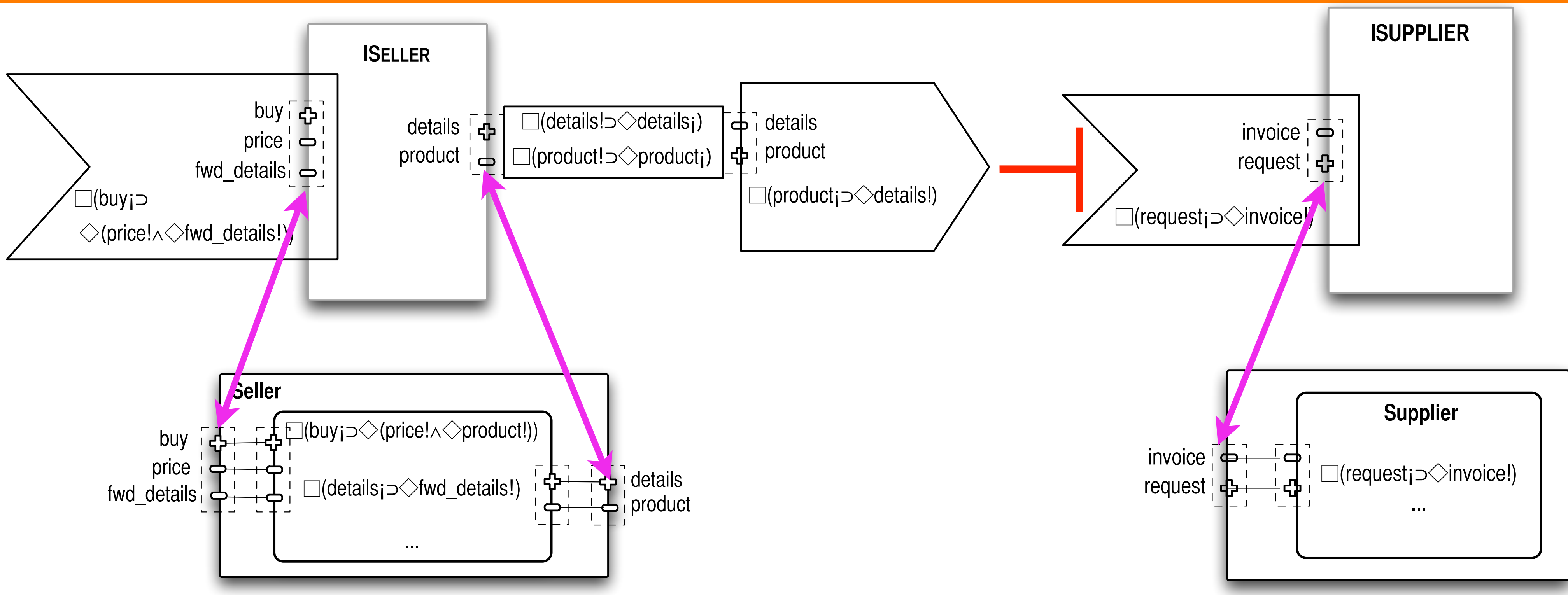


Second theorem – compositionality



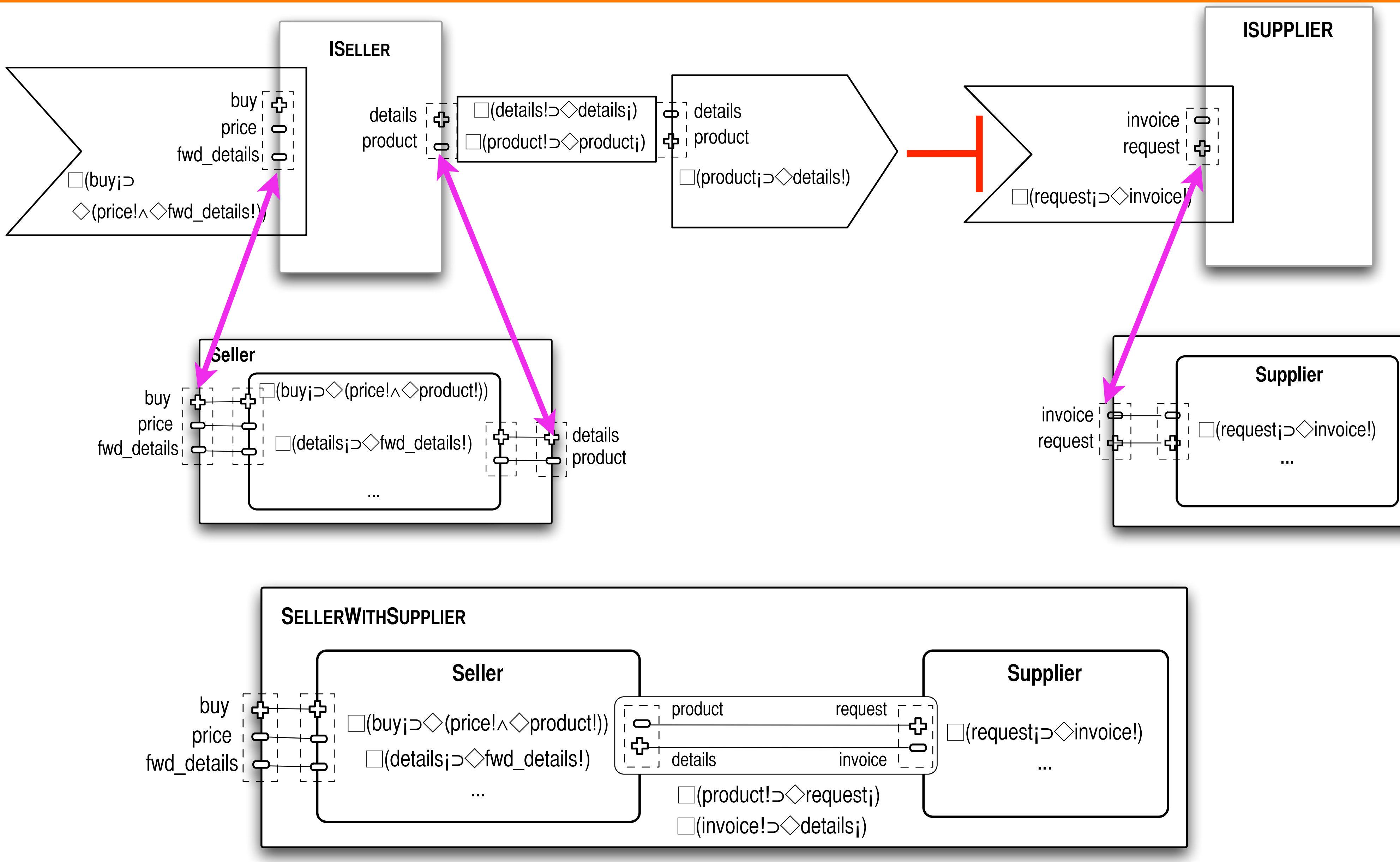
Fiadeiro&Lopes@Aussois2011

Second theorem – compositionality



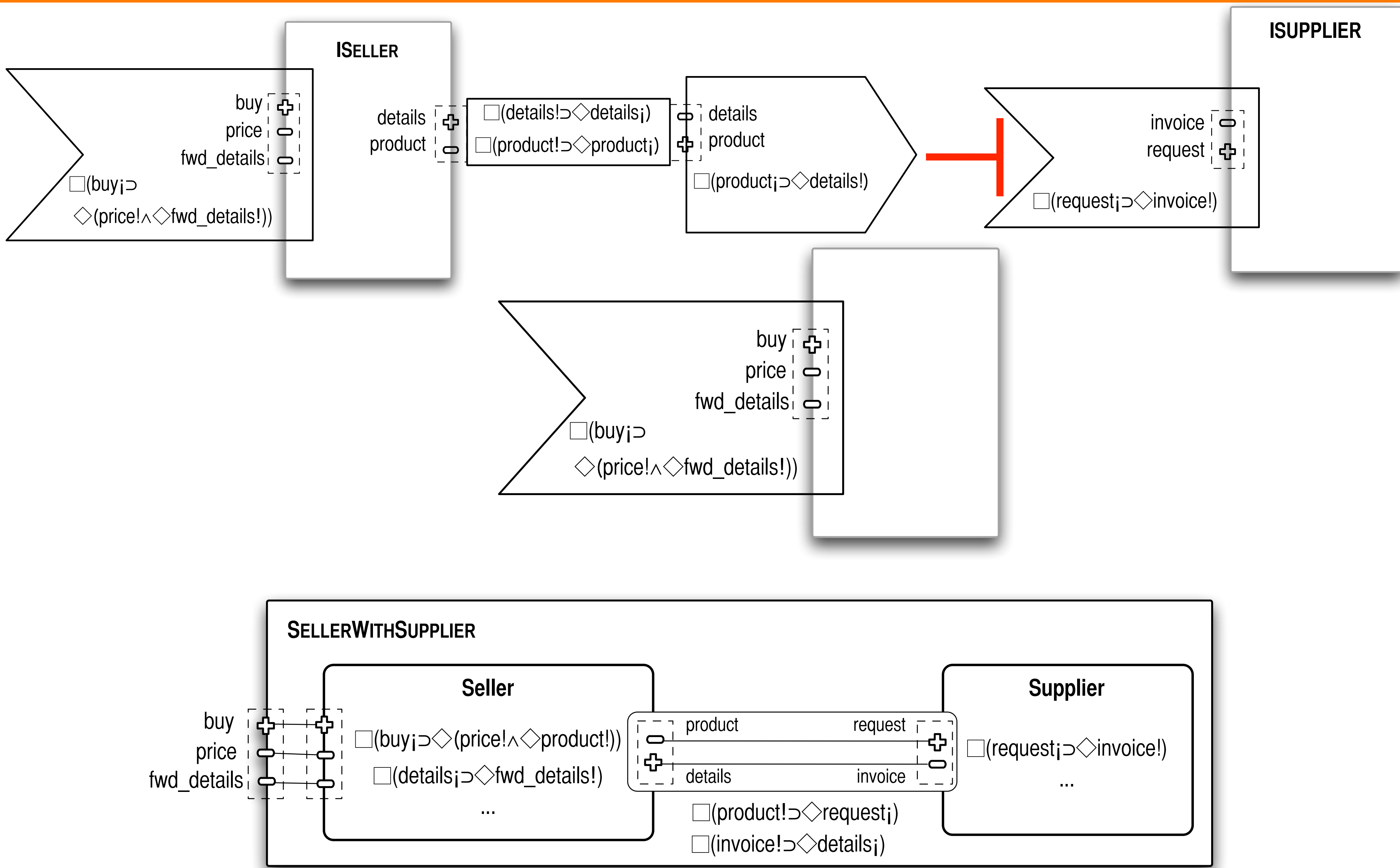
Fiadeiro&Lopes@Aussois2011

Second theorem – compositionality



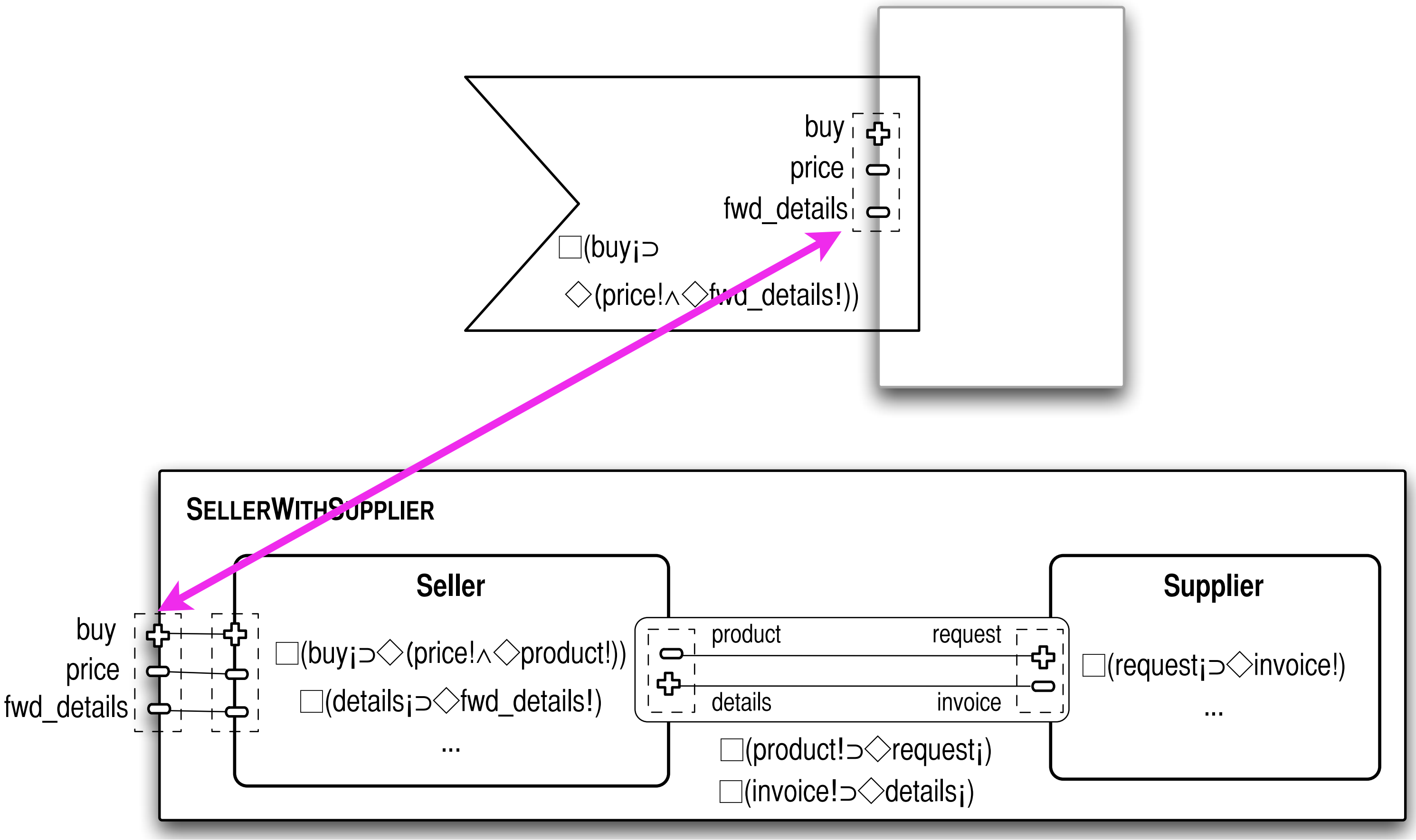
Fiadeiro&Lopes@Aussois2011

Second theorem – compositionality



Fiadeiro&Lopes@Aussois2011

Second theorem – compositionality



Fiadeiro&Lopes@Aussois2011

Thursday, 6 January 2011

Conclusions and further work

- ARNs
  - progress-enabled vs consistency
  - asynchronous model
    - typically, only bounded buffers are required
    - actually, typical business protocols (as in SRML) are finite
  - what is typically unbounded is the ARN (number of processes and channels)

Conclusions and further work

- Dynamic aspects
  - we have developed a model of dynamic discovery and binding (FACJ, ECSA)
  - it needs to be transposed to ARNs
  - and analysed for its theoretical properties