Event-B Patterns and Their Tool Support

Thai Son Hoang, Andreas Fürst and Jean-Raymond Abrial

Department of Computer Science
Swiss Federal Institute of Technology Zürich (ETH Zürich)

SEFM 2009, 23rd-27th November, 2009
Hanoi, Vietnam

(Work supported by DEPLOY, an FP7 European Project)

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Outline

Objective

Objective

Having a systematic method of building systems from re-usable formal models.

This technology allows us:

- to reuse models efficiently, and
- to reduce the effort of doing proofs.
- Such reusable models are called patterns.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Background. Event-B

- Event-B is a notation used for developing mathematical models of discrete transition systems
- Event-B is to be used together with the Rodin Platform
- Such models, once finished, can be used to eventually construct:
  - sequential programs,
  - distributed programs,
  - concurrent programs,
  - electronic circuits,
  - large systems involving a possibly fragile environment,
  - etc.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Background. Event-B Models

Machines and contexts

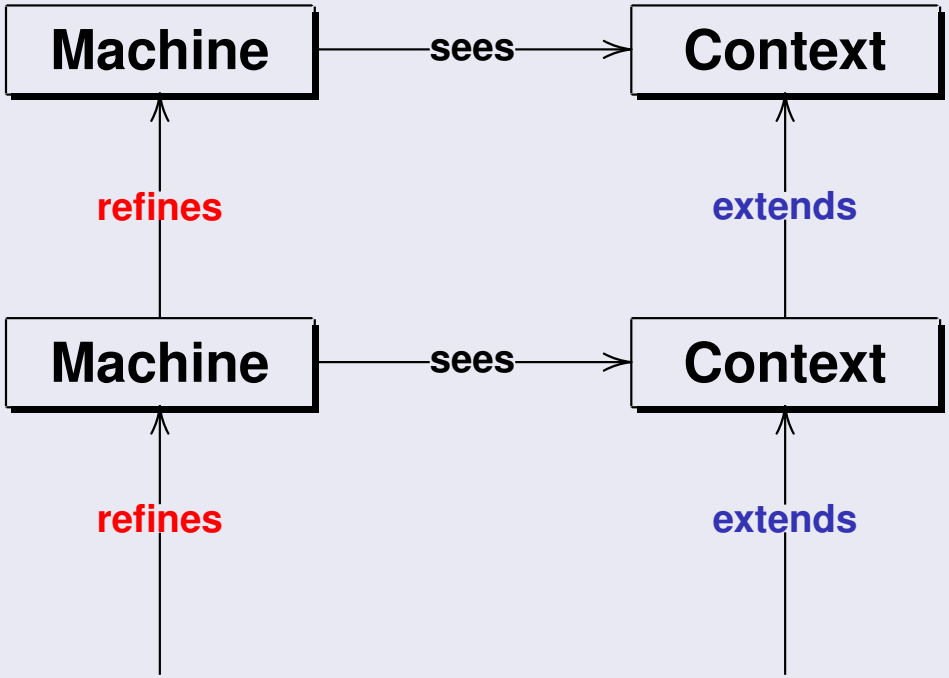
Machine	**Context**
variables	**sets**
invariants	**constants**
events	**axioms**
variant	

- Contexts contain the static part of the model.
- Machines contain the dynamic part of the model.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

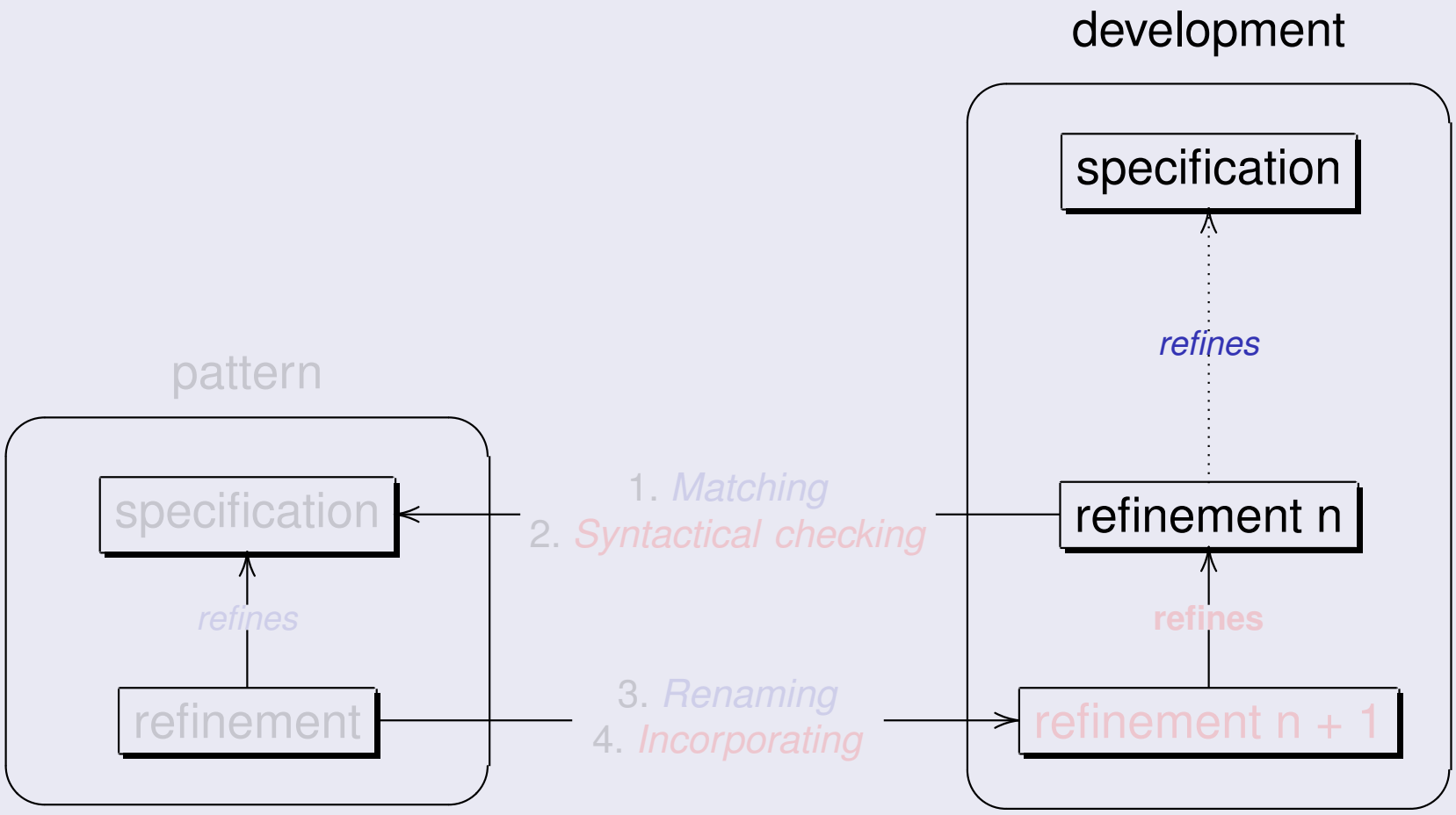
Background. Development Using Refinement

Refinement



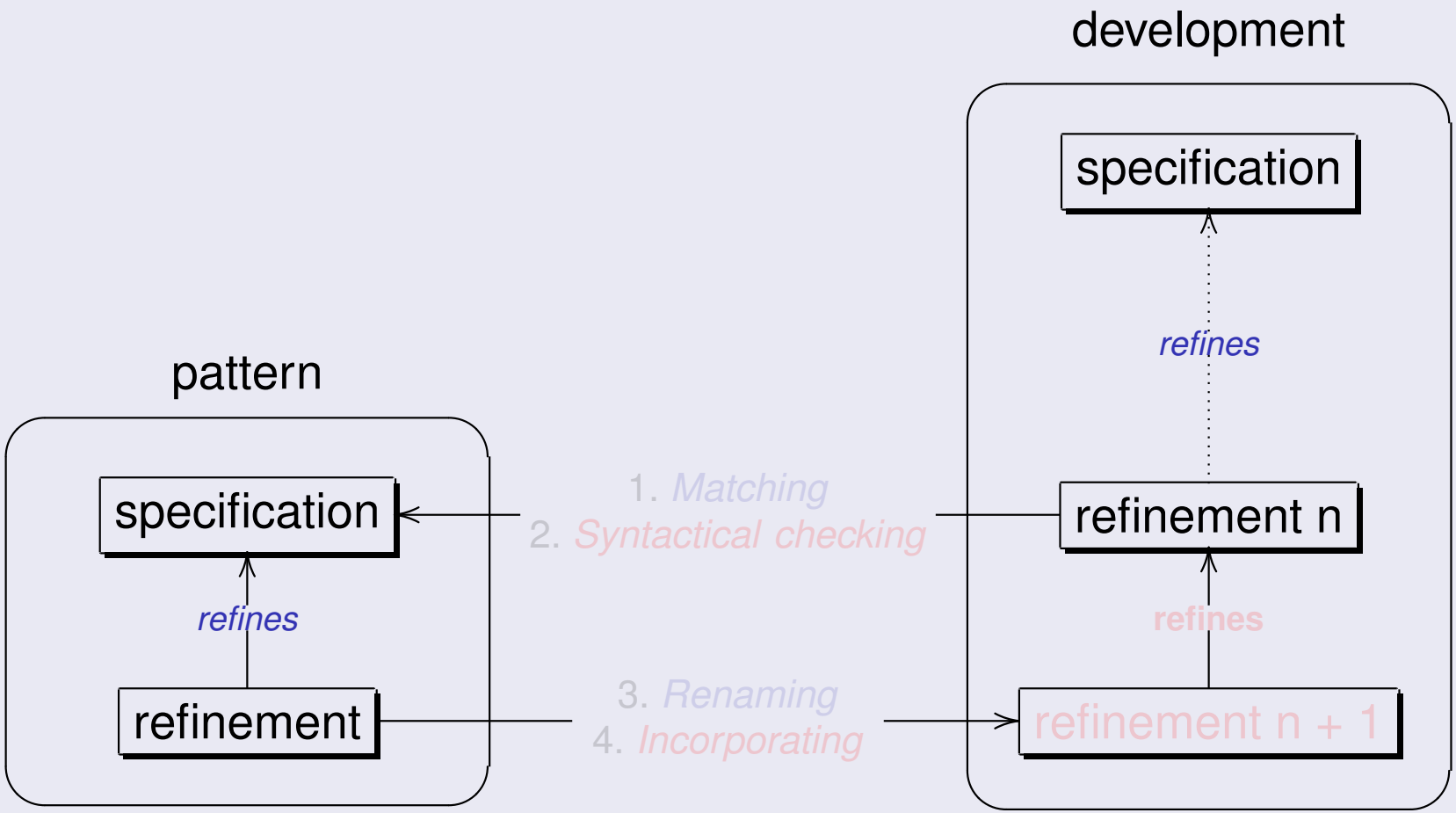
Pattern Incorporation within a Development

High-level view



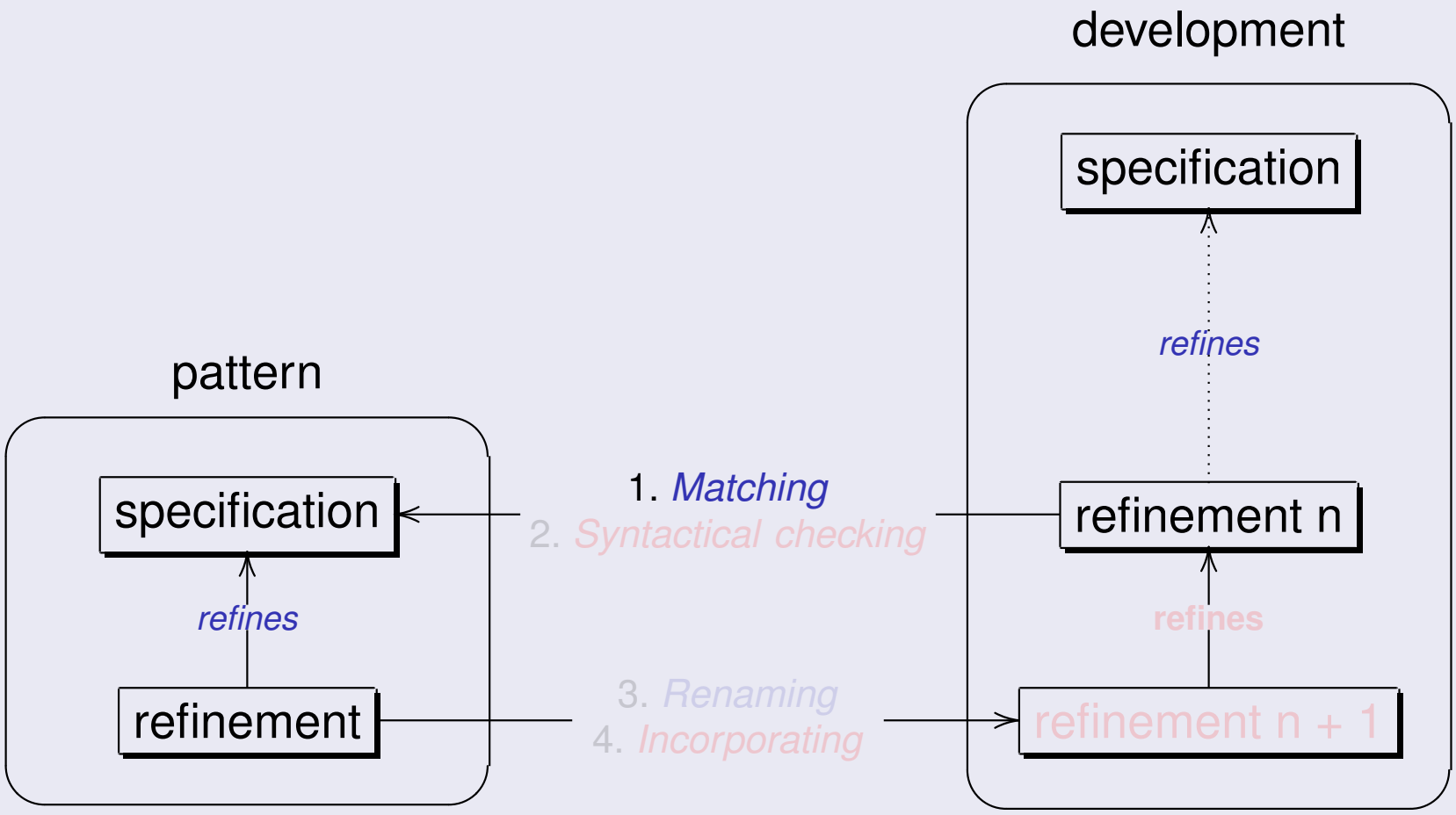
Pattern Incorporation within a Development

High-level view



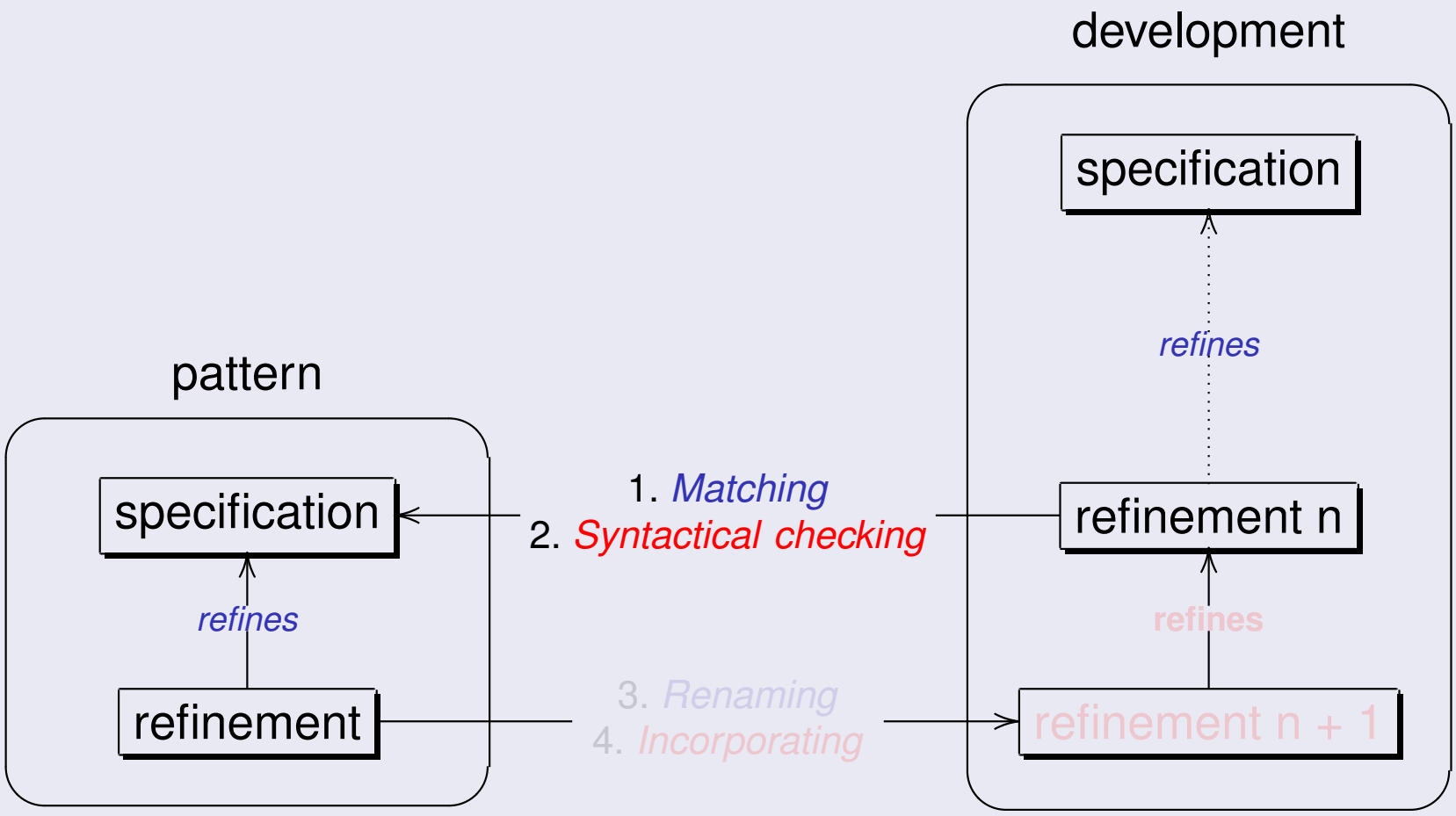
Pattern Incorporation within a Development

High-level view



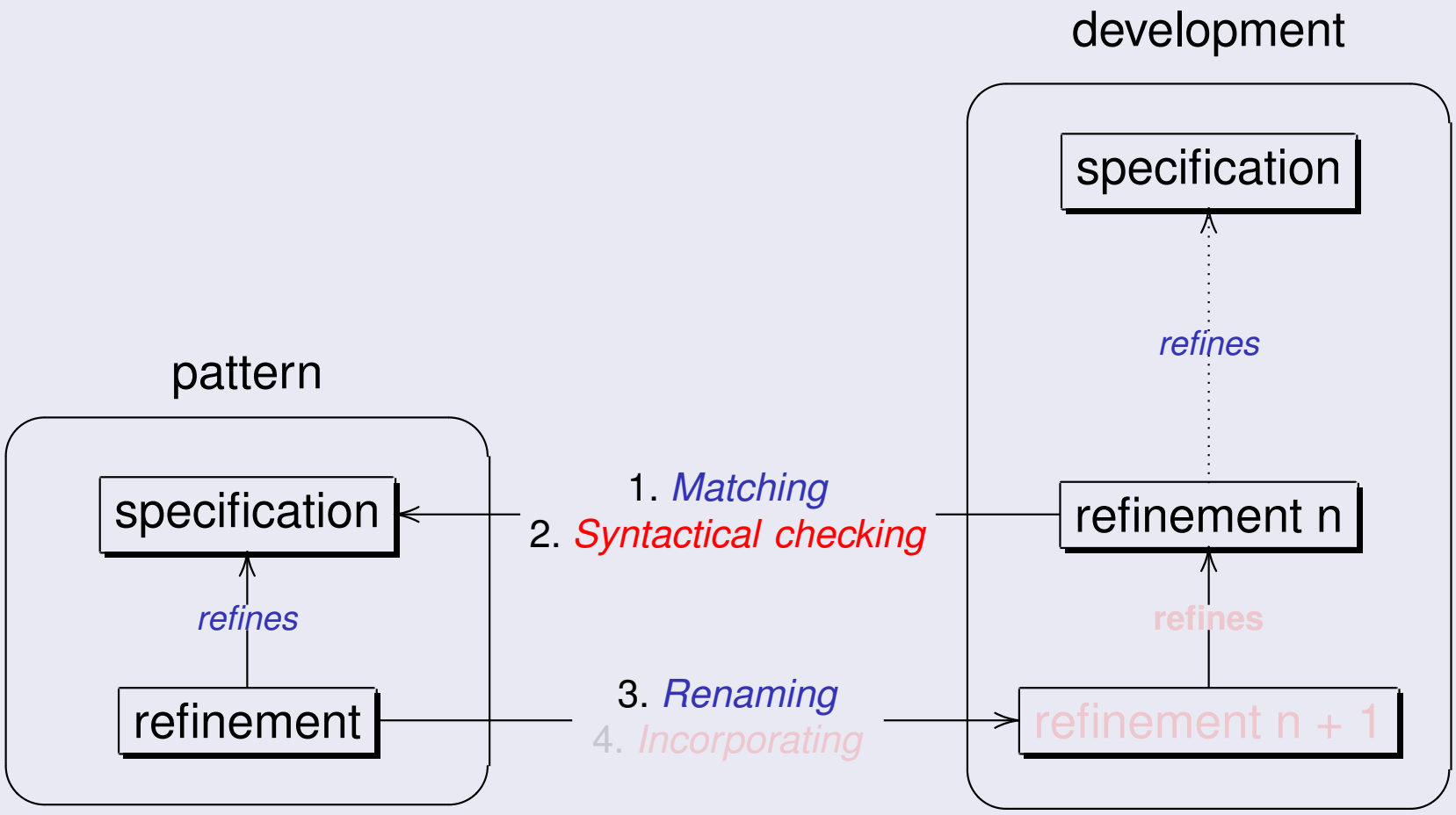
Pattern Incorporation within a Development

High-level view



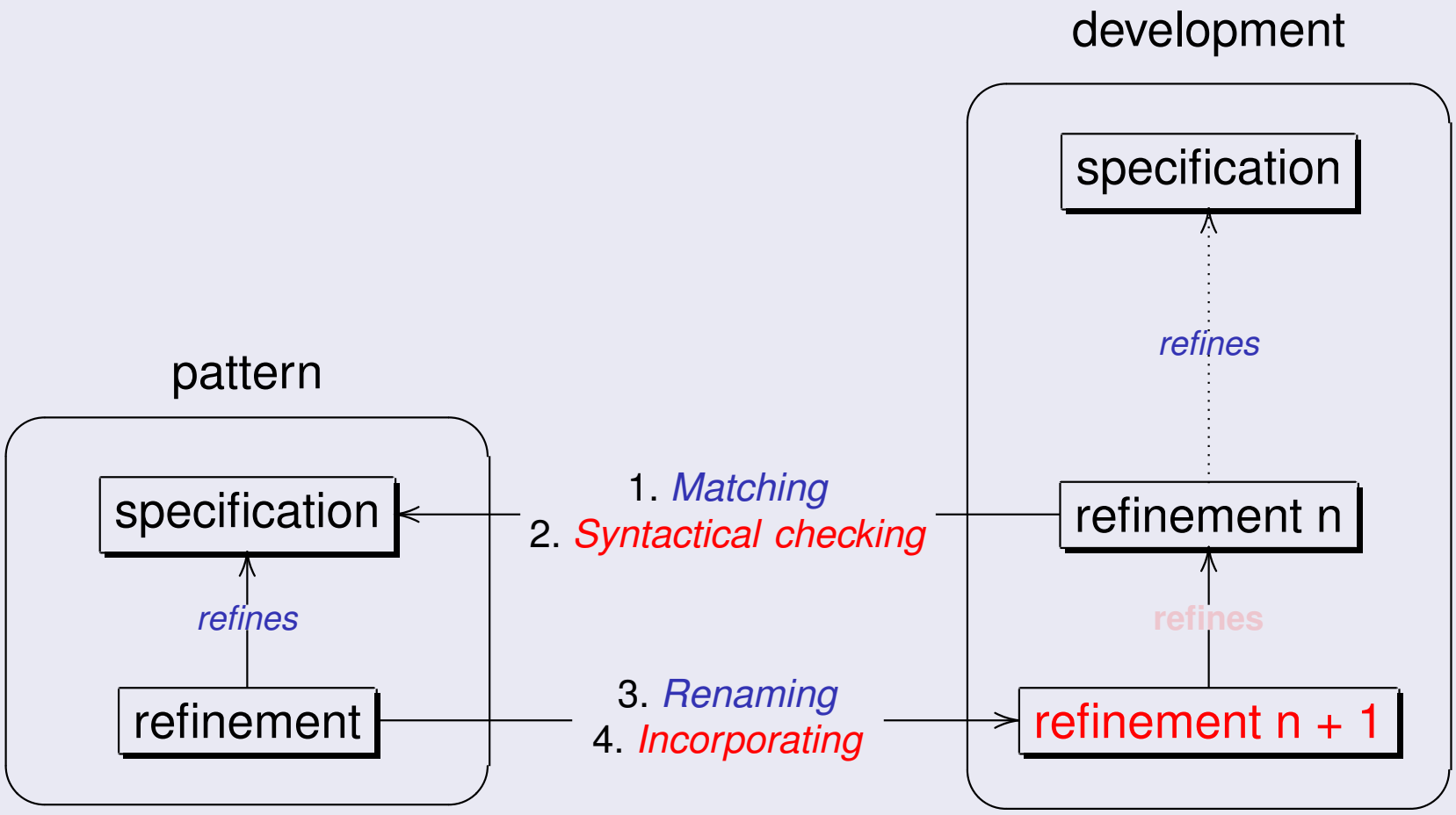
Pattern Incorporation within a Development

High-level view



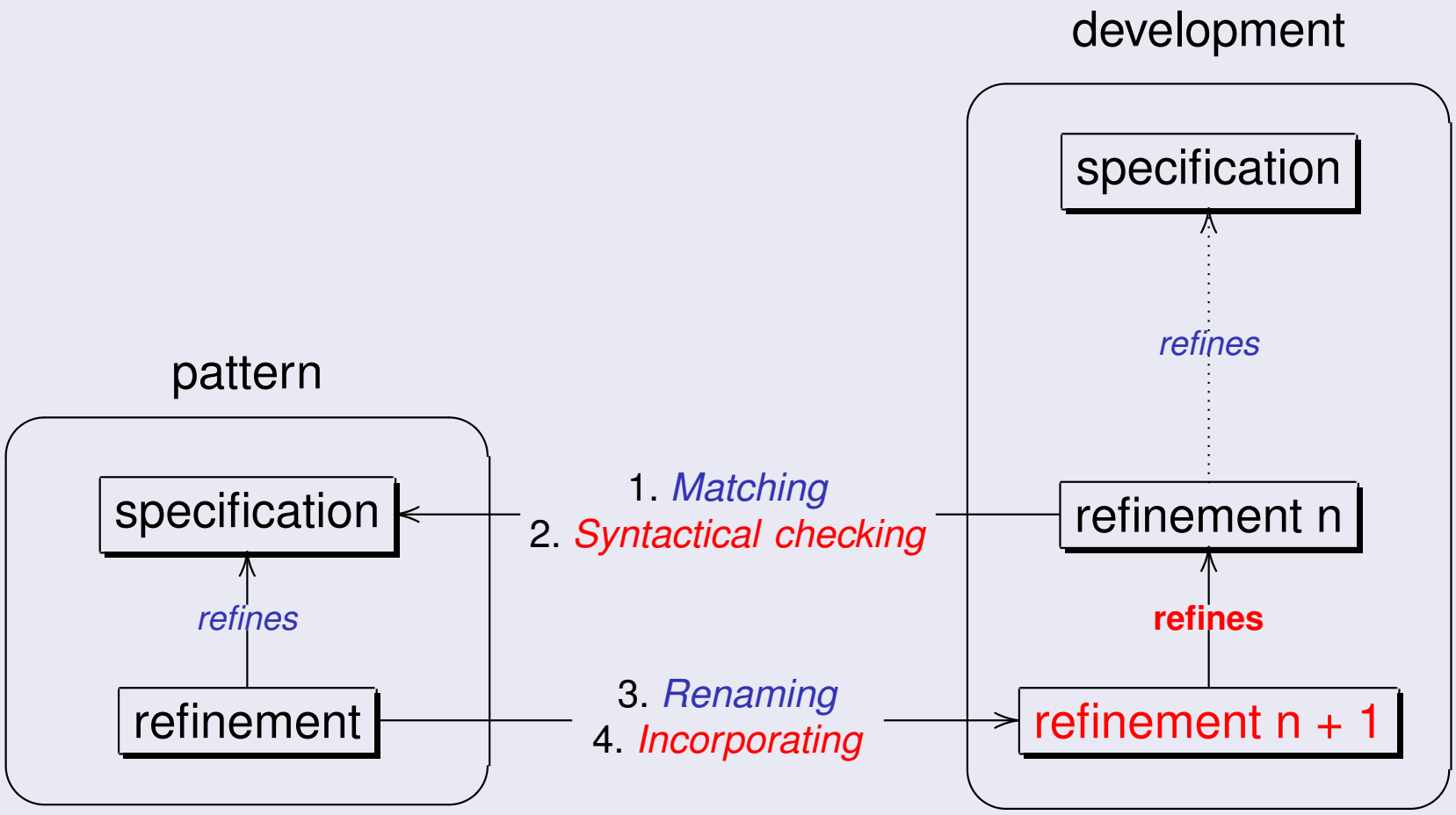
Pattern Incorporation within a Development

High-level view

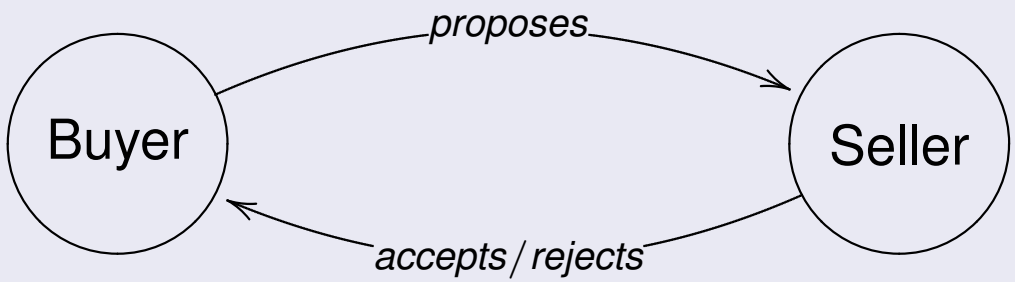


Pattern Incorporation within a Development

High-level view



Problem Description



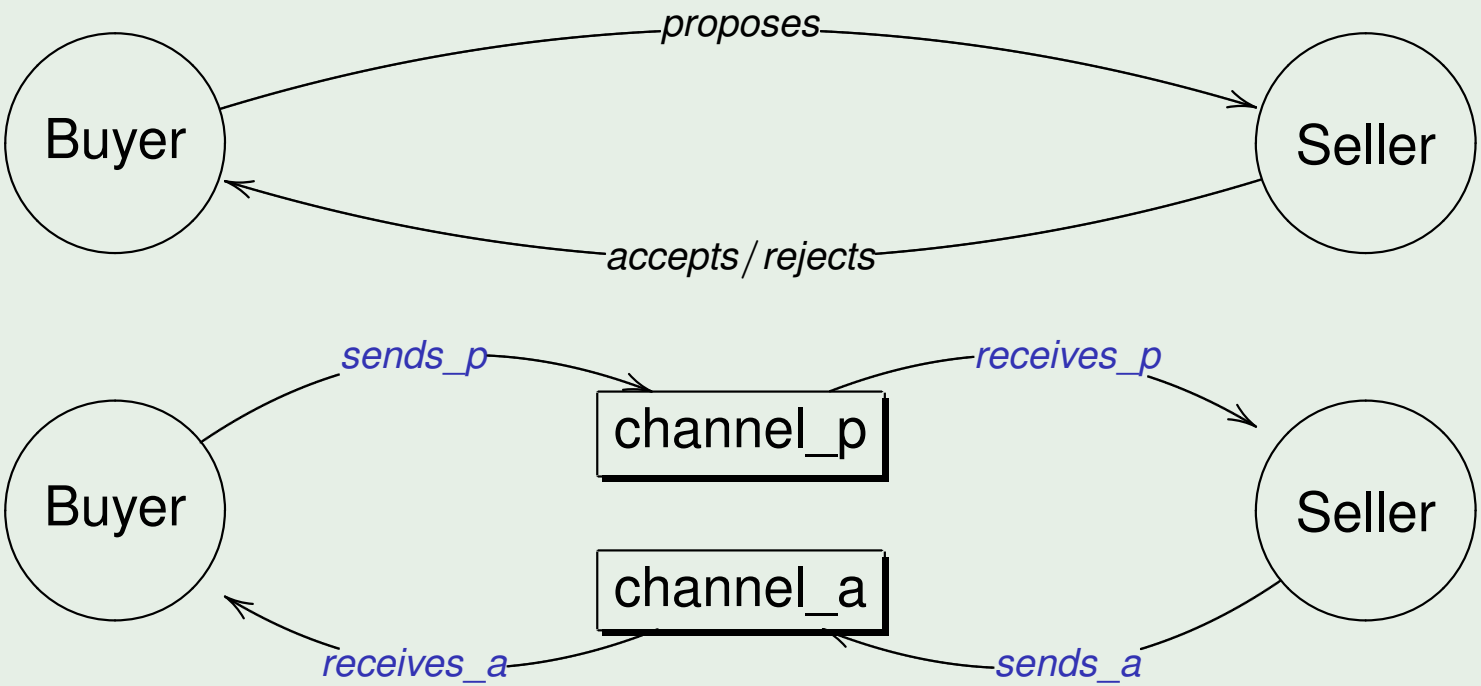
- There are two parties: the *Buyer* and the *Seller*.
- The *Buyer* sends proposals to the *Seller*.
- The *Seller* can either accept or reject a proposal.
- The messages are delivered asynchronously.
- Each site has a separate agreement status:
  either agreed or disagreed.

Property 1

If the *Buyer* and *Seller* agree then they must agree on the same proposal.

Pattern Recognition

Two similar behaviours



We have similar behaviours in sending/receiving proposals and sending/receiving answers (acceptances/rejections), i.e. communication between two parties.

Pattern. Abstract Communication

- We take an abstract view of the communication without the channel.
- There are only two events sends and receives.
- *Demo*: Machine ChannelInterface.

Pattern. Two Different Types of Channels

Communication Properties

- *Demo*: Exactly-One (EO) is in the machine EO.
- *Demo*: Exactly-One-In-Order (EOIO) is in the machine EOIO.

Both of types of channels are refinements of the abstract channel.

Protocol. (1/2)

The *Buyer*

- The *Buyer* can send a proposal any time and set its agreement status to disagree.
- The *Buyer* keeps track of the number of proposals it sends.
- The *Buyer* keeps track of the number of answers (agreement/rejection) it receives.
- The *Buyer* only changes its agreement status to agreed in the case when it receives an agreement and the number of received answers is the same as the number of sent proposals.

Demo: Model of the *Buyer*'s behaviour is in the machine protocol.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Protocol. (2/2)

The *Seller*

- The *Seller* answers to all proposals that it receives.
- The *Seller* set its agreement status to agreed when sending acceptance message.
- The *Seller* set its agreement status to disagreed when sending rejection message.

Demo: Model of the *Seller*'s behaviour is in the machine protocol.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Our Experiment

Question

Which type of channel will maintain Property 1.

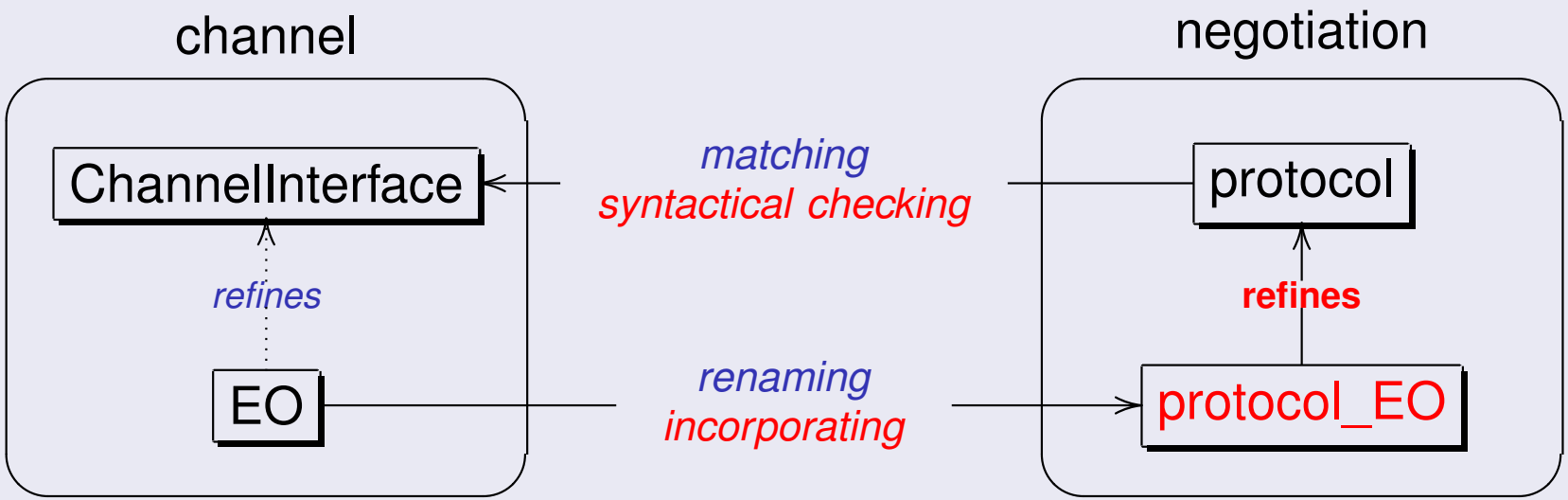
Property 1 (again)

If the *Buyer* and *Seller* agree then they must agree on the same proposal.

Experiment with EO

Exactly-One Channel

We apply our pattern approach with the “Exactly-One” channel.



Here, we “instantiate” the channel pattern twice:
for *proposals* and for *answers*.

Experiment with EO. Step 1/4

Matching

Usage 1

MESSAGE	$\rightsquigarrow$	*PROPOSAL*
B_sent_proposal_count	$\rightsquigarrow$	*sent_count*
S_received_proposal_count	$\rightsquigarrow$	*received_count*
B_sends_proposal	$\rightsquigarrow$	sends
S_receives_proposal	$\rightsquigarrow$	receives

Usage 2

MESSAGE	$\rightsquigarrow$	*BOOL*
S_sent_answer_count	$\rightsquigarrow$	*sent_count*
B_received_answer_count	$\rightsquigarrow$	*received_count*
S_sends_rejection	$\rightsquigarrow$	sends
B_receives_rejection	$\rightsquigarrow$	receives
S_sends_acceptance	$\rightsquigarrow$	sends
B_receives_acceptance_right	$\rightsquigarrow$	receives
B_receives_acceptance_wrong	$\rightsquigarrow$	receives

Experiment with EO. Step 2/4

Syntactical Checking

We need to check if the events are matched with each other.

sends
  any *msg* **where**
    $msg \in MESSAGE$
  then
    $sent\_count := sent\_count + 1$
  end

B_sends_proposal
  any *prop* **where**
    $prop \in PROPOSAL$
  then
    $B\_last\_sent\_data := prop$
    $B\_agreed := FALSE$
    $B\_sent\_proposal\_count := B\_sent\_proposal\_count + 1$
  end

Hochschule Zürich
echnology Zurich

Experiment with EO. Step 3/4

Renaming

Renaming the variables of the pattern refinement before incorporation.

Usage 1

$$\begin{array}{lcl} channel & \rightsquigarrow & B2S\_channel\_proposal \\ sent & \rightsquigarrow & B\_sent\_proposals \\ received & \rightsquigarrow & S\_received\_proposal \end{array}$$

Usage 2

$$\begin{array}{lcl} channel & \rightsquigarrow & S2B\_channel\_answer \\ sent & \rightsquigarrow & S\_sent\_answers \\ received & \rightsquigarrow & B\_received\_answers \end{array}$$

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Experiment with EO. Step 4/4

Incorporation

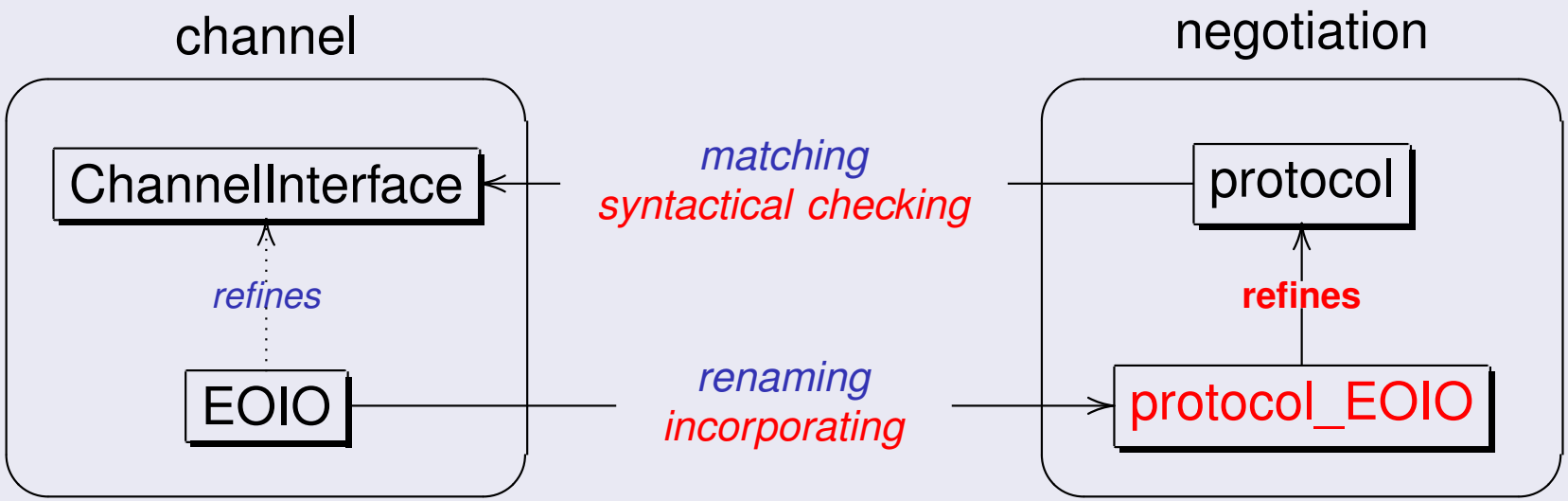
- We incorporate the pattern refinement into the development to create the refinement of the protocol with the EO channel.
- *Demo*: The result is in the machine protocol_EO

Experiment with EO. Preservation of the Property 1

- The EO channel protocol does not maintain Property 1.
- Proposals can be re-ordered while transferring to the seller.
- The invariant cannot be proved to be maintained.

Experiment with EOIO

A similar experiment



- The result after incorporation is in the machine protocol_EOIO.
- The EOIO channel protocol does maintain Property 1.
- The invariant can be proved.

Statistics

Statistics

	Total	Automatic	Manual
EO pattern	20	13	7
EOIO pattern	23	18	5

Table: Proof statistics

	Total	Automatic	Manual
Without patterns	83	59	24
With patterns	17	13	0
Saving	66 (80%)	42 (71%)	24 (100%)

Table: Proof statistics. Protocol EO

	Total	Automatic	Manual
Without patterns	110	93	17
With patterns	36	33	3
Saving	74 (67%)	60 (65%)	14 (82%)

Table: Proof statistics. Protocol EOIO

Hochschule Zürich
echnology Zurich

Tool Support Requirements

A prototype

The prototype has been built as a plug-in for the Rodin Platform.

1. Matching of the variables and events.
   (done by developers, supported by tool dialogue)
2. Syntactical checking of the guards and actions.
   (automatically by the tool)
3. Renaming of the variables and events, if necessary.
   (done by developers, supported by tool dialogue)
4. Incorporating of the pattern refinement into the main development.
   (automatically by the tool)

Conclusions and Future Work

- An approach for reusing formal models (including design decisions).
- Tool support is implemented as a plug-in for Rodin Platform.

- Provide link to generic instantiation (instead of renaming).
- Graphical input for the pattern application.
- Applying the approach to other domains.