B-Tropos :

Agent-oriented requirements engineering meets computational logic for declarative business process modelling and verification

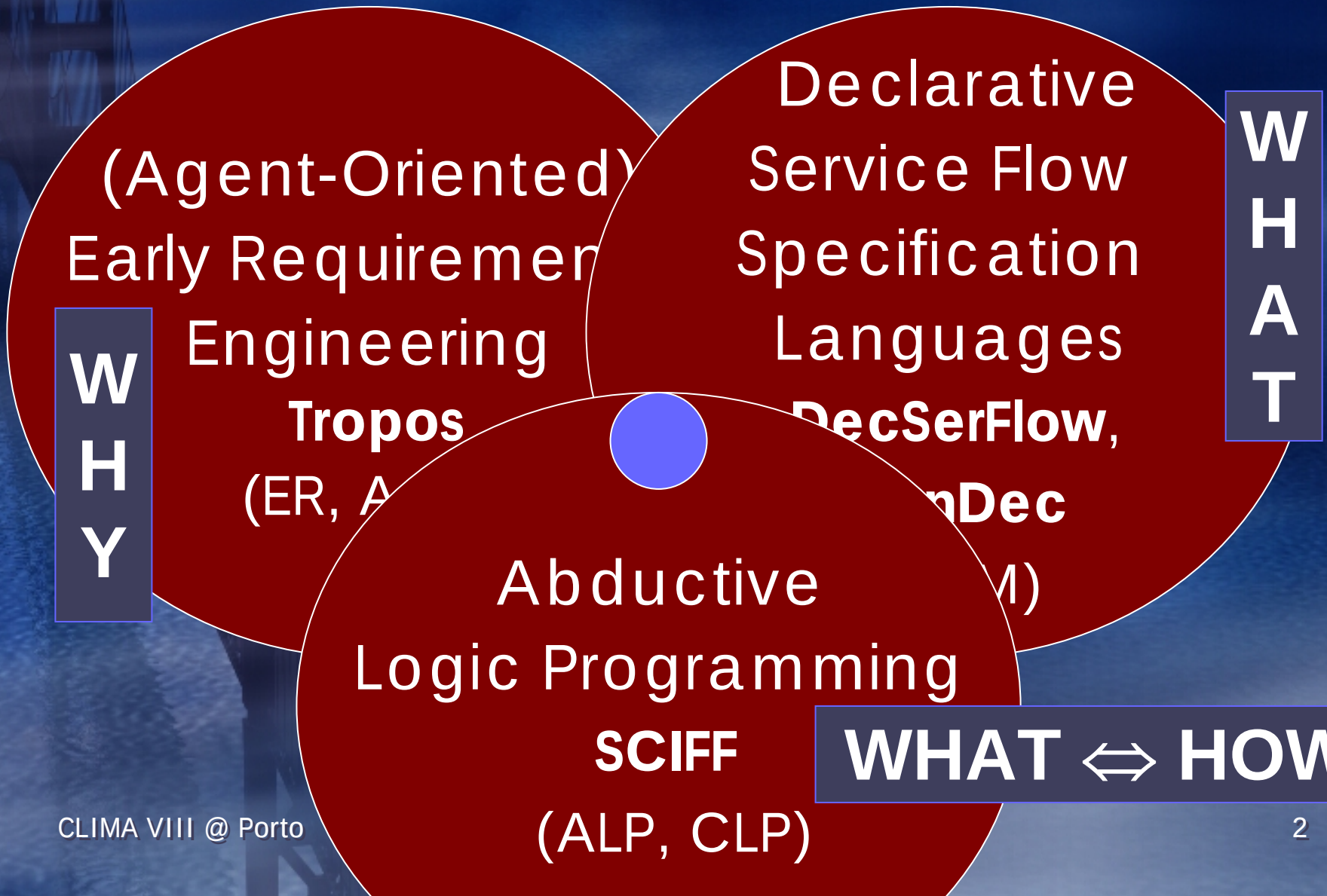
Paolo Torroni, U. Bologna

Joint work with

Volha Bryl, Nicola Zannone, U. Trento

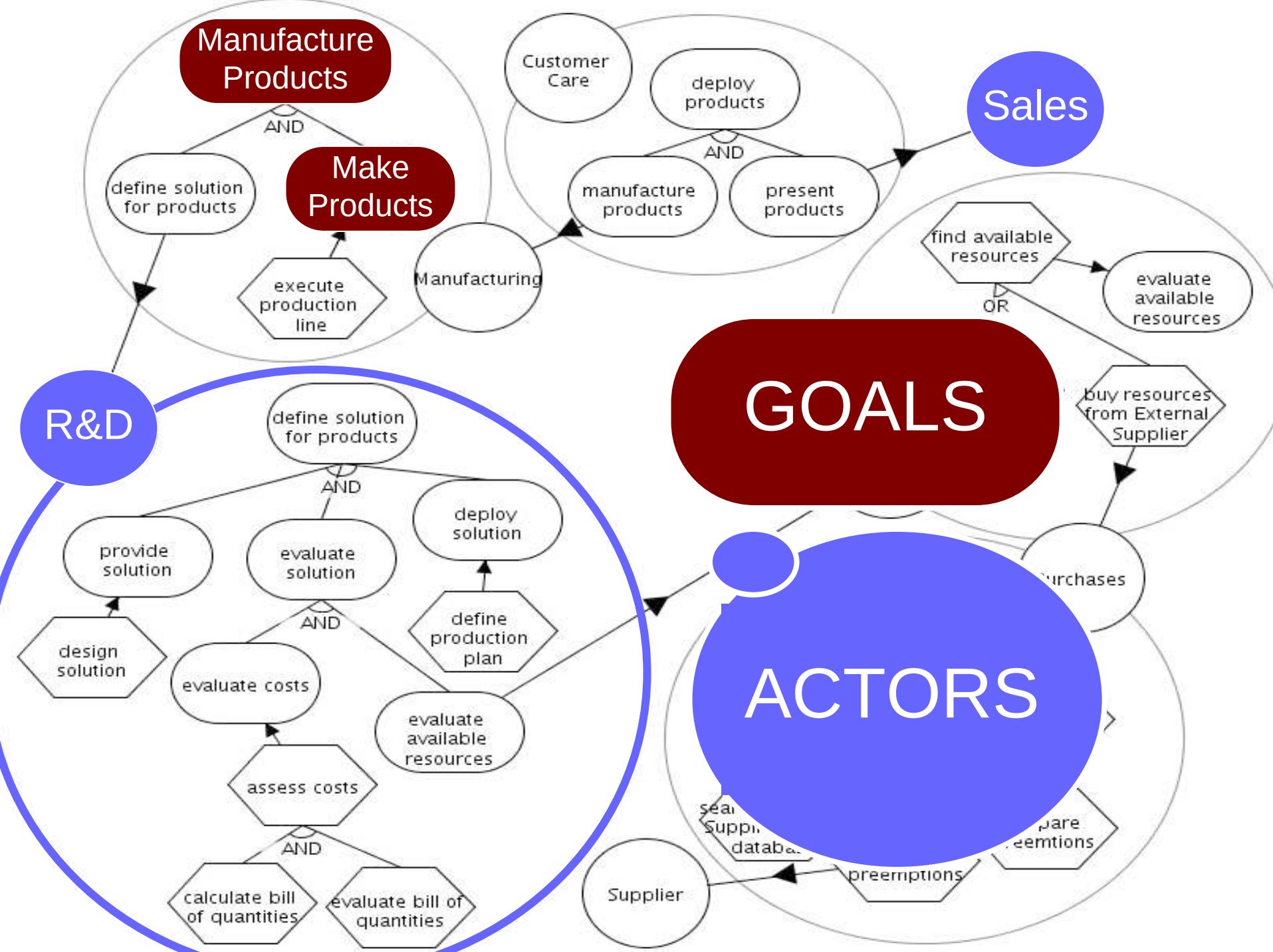
Marco Montali, Paola Mello, U. Bologna

Positioning



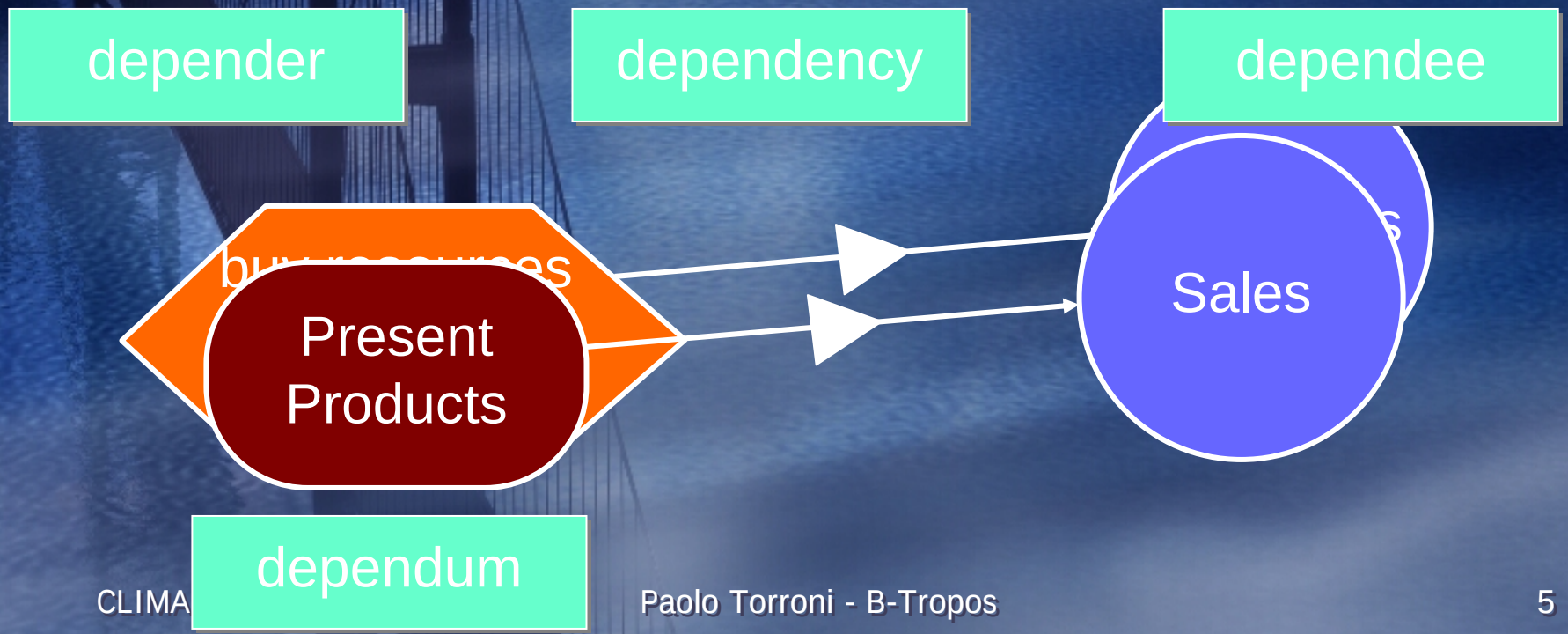
Overview

- Tropos
- B-Tropos
- Mapping B-Tropos onto SCIFF
- Conclusion


Manufacture Products
AND
define solution for products
Make Products
execute production line
Manufacturing
Customer Care
deploy products
AND
manufacture products
present products
Sales
find available resources
OR
evaluate available resources
buy resources from External Supplier
GOALS
R&D
define solution for products
AND
provide solution
evaluate solution
deploy solution
design solution
AND
define production plan
evaluate costs
evaluate available resources
assess costs
AND
calculate bill of quantities
evaluate bill of quantities
ACTORS
Supplier
preemptions

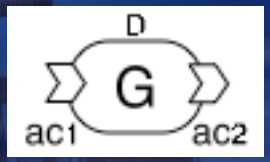
Actor diagram

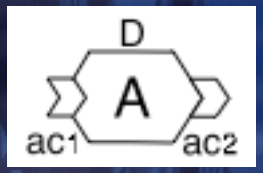
- Stakeholders and system actors
- Network of dependencies



Goal diagram

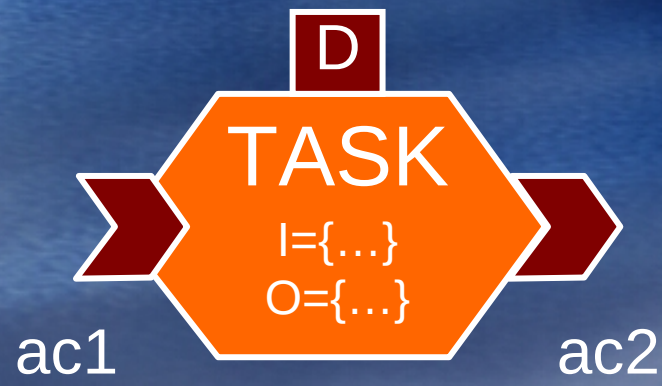
- Perspective of a specific actor
- Analysis of actor's goals
  - Means-end analysis
  - AND/OR decomposition
  - Contribution analysis



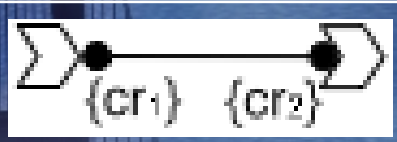
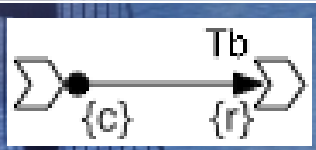
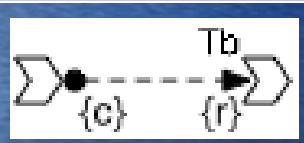
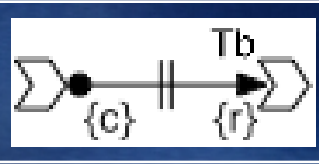
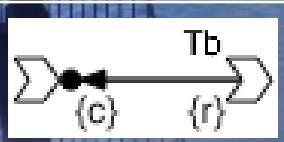
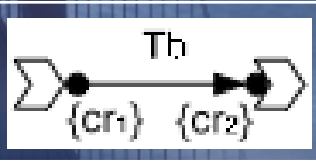
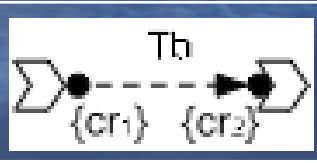
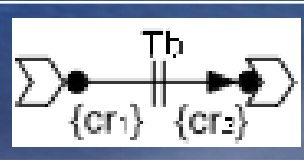


Extended notation

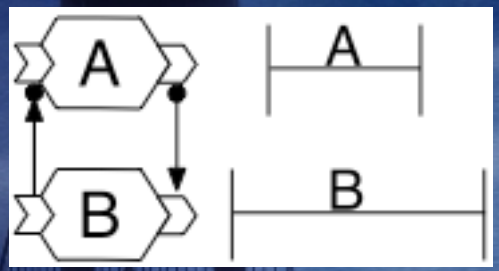
- Start & Completion
- Min/max duration time interval
- Input/Output data
- (Time) Constraints

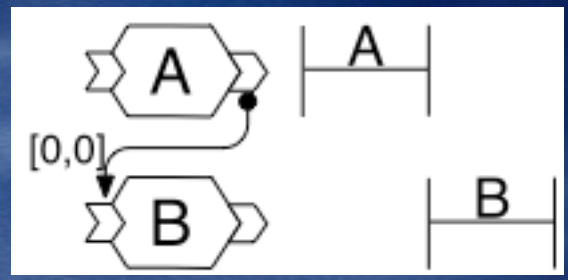


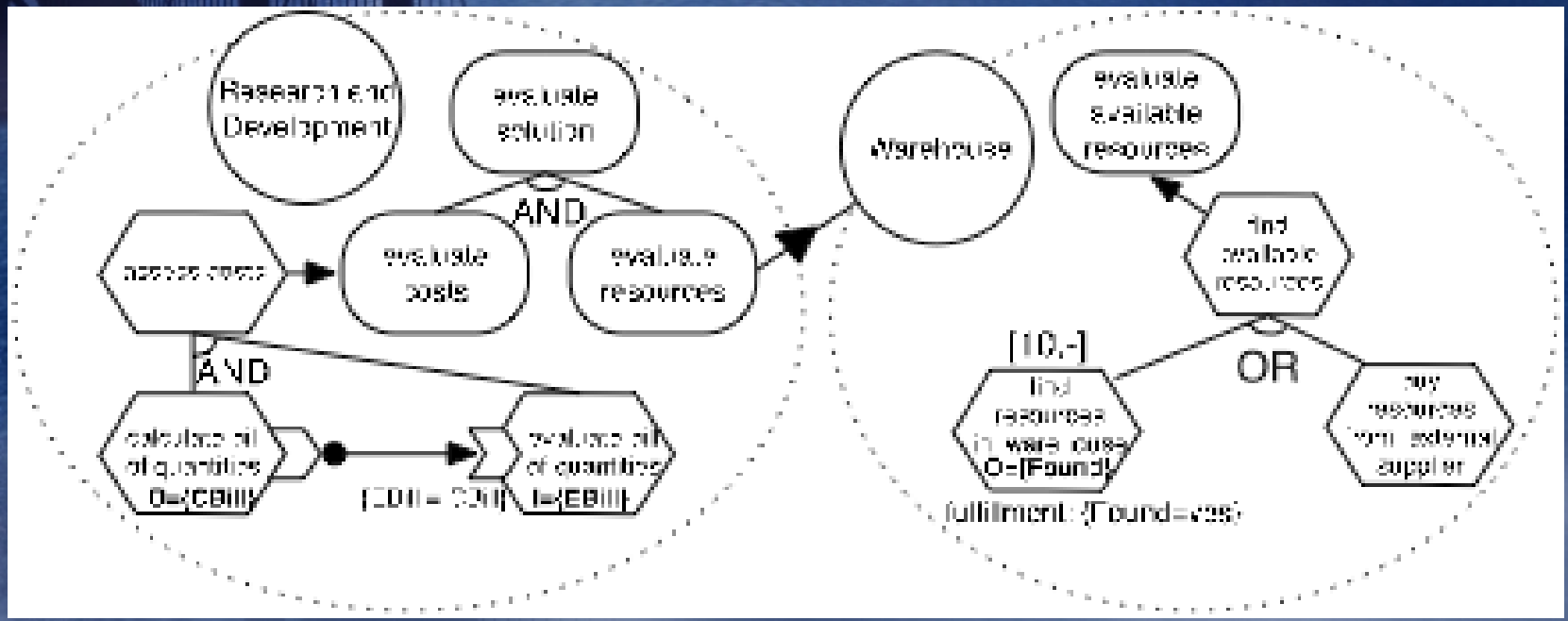
Process-oriented constraints (Tropos meets DecSerFlow)

	relation	weak relation	negation
responded presence			
co-existence			
response			
precedence			
succession			

Examples


A
A
B
B


A
A
[0,0]
B
B


Research and Development
evaluate solution
AND
evaluate costs
evaluate costs
evaluate resources
AND
calculate all of quantities
0=[CBill]
evaluate all of quantities
[CBill = EBill]
[EBill]
Warehouse
evaluate available resources
find available resources
[10,-]
find resources in ware ouse
0=[Found]
OR
any resources from external supplier
fulfilment: [Found=yes]


From modeling to reasoning
B-Tropos meets SCIFF

Abductive Logic Programming

- Starting from a goal G, typically not entailed by a given knowledge base: $KB \not\models G$,
- Find a set $\Delta \subseteq A$ such that
  - G is entailed by the KB
  - The IC are not violated

$$KB \cup \Delta \models G$$
$$KB \cup \Delta \models IC$$

- Bonus: Δ is “minimal”

SCIFF: ALP language & p-p

- **Events/expectations** are first-class citizens
- Nice integration with **CLP** (constructive abduction/negation)
- Good theory developed (with strong results)
- Thoroughly tested implementation
  - Uses constraints technology
  - Relatively efficient (tested with thousands of events, gives results within seconds/minutes)
  - Nice graphical interface and many add-ons
- Can reason on run-time upcoming events
  - use same language also for analysis/monitoring/checking/...
- And more

SCIFF Syntax

- SCIFF program: $\langle$ ***KB***, ***A***, ***IC*** $\rangle$

- ***KB***: set of definitions Head $\leftarrow$ Body
- ***A***: abducible atoms (including **E**,**EN**, and **H**)
- ***IC***: set of integrity constraints Body $\rightarrow$ Head
  - Expressions that *must be true at all times*
  - Head ::= Disjunct $\vee$ Disjunct $\vee$ ... $\vee$ Disjunct
- ***Goal***: conjunction of atoms and constraints

SCIFF: Events & Expectations

H(Ev, T)

E(Ev, T)

EN(Ev, T)

SCIFF: Declarative Semantics

- Extends semantics of ALP

$$KB \cup \mathrm{E} \cup \Delta \models G$$
$$KB \cup \mathrm{E} \cup \Delta \models IC$$

- By defining (minimal) abductive answers

$$\langle \mathrm{E}, \Delta \rangle$$

representing possible ways to achieve G

From B-Tropos to SCIFF

- AND/OR decompositions (trivial)
- Positive contributions: target achieved if contribution's source is achieved
- Negative contributions: denials
- Goal/task dependencies: expectation on depender's behaviour
- Underspecified models: use abduction

Goals & Tasks

- achieve(ACTOR,GOAL,T_i,T_f) ←
  $T_f \in D^{+Ti} \wedge ac_1 \wedge ac_2 \wedge \ldots$
- execute(ACTOR,TASK,T_i,T_f) ←
  $T_f \in D^{+Ti} \wedge ac_1 \wedge ac_2 \wedge \ldots$

Task & goal dependency

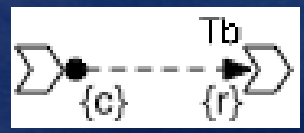
- execute(wareh,buy_res_e_s,T_i,T_f) ← **E**(delegate(wareh,purchases,buy_res_e_s,T_f), T_i).
- **H**(del(customer_care,sales,present_prod,T_f), T_i) → achieve(sales,pres_prod,T_i,T_f) ∧ $T_i > T_d$

Response connections

- Response

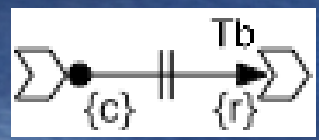
hap(event(Ev,A,X),T_1) $\wedge$ c
$\rightarrow$ **exp**(event(Ev,A,X),T_2) $\wedge$ r $\wedge$ $T_2 \in T_b^{+T1}$

- Weak response



hap(event(Ev,A,X),T_1) $\wedge$ c $\wedge$ **hap**(event(Ev,A,X),T_2)
$\rightarrow$ r $\wedge$ $T_2 \in T_b^{+T1}$

- Negation response



hap(event(Ev,A,X),T_1) $\wedge$ c
$\wedge$ **hap**(event(Ev,A,X),T_2) $\wedge$ r $\wedge$ $T_2 \in T_b^{+T1} \rightarrow \perp$

Results so far (ongoing)

- Process-oriented constraints in Tropos
- Mapping of (very) high-level concepts into LP framework
- Modeling of sample scenarios (taken from national FIRB project)
- Verification of model properties and properties of mapping (termination)
- Possible animation using SCIFF agents
- Run-time verification of compliance (e.g. delegation) *using the same language*

Conclusion

- Research being done in the intersection of three areas (still ongoing)
- Touches three important aspects (why, what, how)
- Role of LP is to perform reasoning (verification)
- Graphical notation extended (B-Tropos)
- Focus on temporal reasoning
- Implementation! Being integrated in Tropos