Lecture III
Social Dependency Models in RE

John Mylopoulos
University of Ottawa

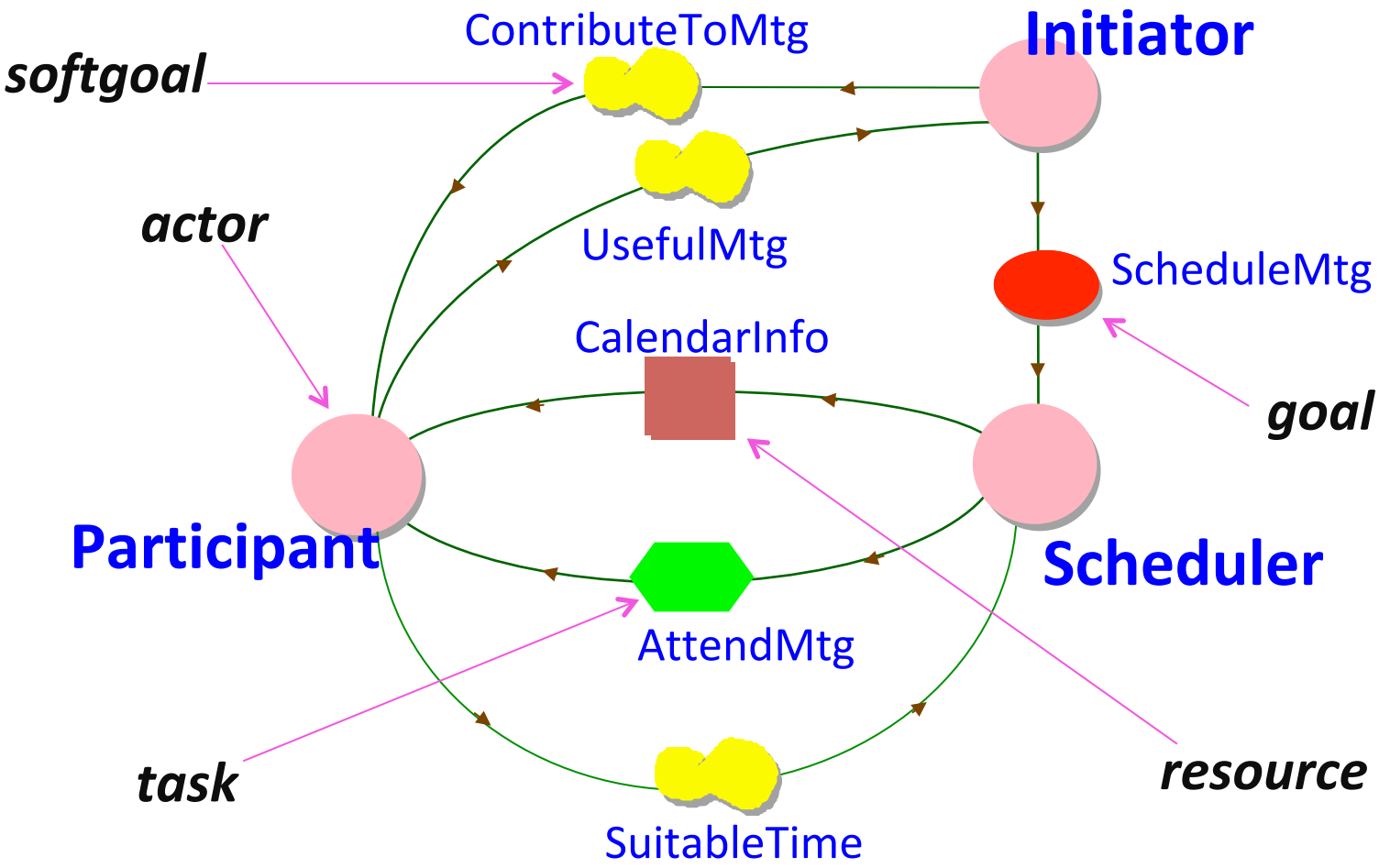
Federal University of Pernambuco (UFPE),
Recife, November 18, 2019

Social dependency models

- Social dependence models assume an ontology of actors (agents, roles) and social dependencies among them.
- *i** pioneered such models with its actor dependency diagrams [Yu95].
- But there were other proposals in the same period: Munindar Singh proposed ***social commitments*** as fundamental concepts for multi-agent systems [Singh91], [Singh99], while in Law, ***rights*** and ***obligations*** have been acknowledged as foundational legal (and therefore social) concepts [Hohfeld13].
- We review such concepts and how they have been adopted and used in RE.

Social dependencies in *i**

- Actor dependence models consist of actors (roles, agents), and dependencies (goal, softgoal, task, resource) among them.



Formalizing social dependencies

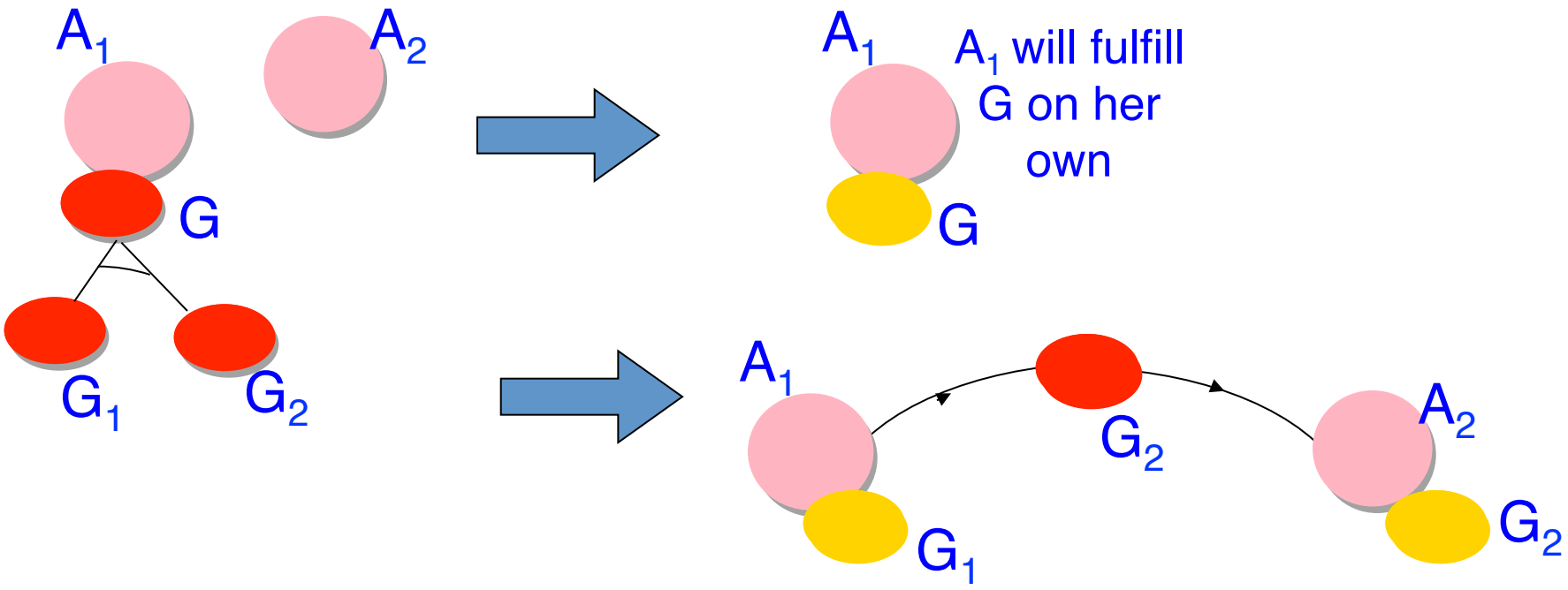
- Actor dependencies are formalized in Eric Yu's PhD thesis using intentional concepts, such as ***beliefs***, ***goals*** and ***commitments***.
- For example, here are the axioms for ***committed goal*** dependency:

$$CW(a, b, \phi) \equiv CW(a, \phi) \wedge B(a, CA(b, \phi))$$
$$CW(a, \phi) \Rightarrow B(a, \exists p, \exists \phi_0(\neg \phi \Rightarrow fail(a, p, \phi_0)))$$
$$CA(a, \phi) \Rightarrow B(a, \exists p(result(p, \phi) \wedge allDepOK(a, p)))$$

- AllDepOK includes all the constraints that need to apply for actor a to execute plan p.
- The formalization of CW (Committed Want), CA (Committed Able) were adopted from Yves Lesperance's PhD thesis.

The delegation problem

- The delegation problem takes as input a collection of actors and their goals, and selects a suitable network of dependencies for fulfilling all actor goals, given constraints on what dependencies are allowed.



Solving the delegation problem

- The delegation problem can be solved with an AI planner.
- But we need to specify more details for any one problem:
  - ✓ Type(g : goal; gt : gtype) – goals can have a type
  - ✓ Order(g1 : goal; g2 : goal) – fulfill g1, then g2
  - ✓ CanSat(a : actor; g : goal) – a can solve g
  - ✓ CanSatT(a : actor; gt : gtype) – a can solve gt goals
  - ✓ Wants(a : actor; g : goal) – a wants g
  - ✓ CanDep(a1 : actor; a2 : actor) – a1 can depend on a2
  - ✓ CanDep4gt(a1: actor; a2 : actor; gt : gtype) – a1 can depend on a2 for goals of type gt
  - ✓ CanDep4g(a1: actor; a2: actor; g: goal) -- a1 can depend on a2 for g.

Experimental evaluation

- Bryl used three criteria for selecting the best plan:
  - ✓ Number of actions in a plan
  - ✓ Overall plan cost
  - ✓ Degree of satisfaction of quality requirements
- The LPG-td planner was used in her experiments [LPG].
- The planner can solve problems involving <10 actors and up to 25 goal graphs of modest size.

Social commitments

- A commitment **C(debtor, creditor, antecedent, consequent)** is a promise from a ***debtor*** to a ***creditor*** to achieve the ***consequent*** if the ***antecedent*** holds.
- The debtor and creditor are actors, antecedent, consequent are propositions describing states-of-affairs.
- Examples:
  - ✓ C(Ebook, Alice, $12payment, BraveNewWorld)
  - ✓ C(Amit, John, vacation, lecture-on-commitments)
  - ✓ C(Alice, Barbara, goodWeather, goOnTrip)
  - ✓ C(UniTN, Fabiano, passExams, getDegree)

[Credits for these notes on commitments: Amit Chopra]

Commitments are everywhere …

- Examples:
  - ✓ A book loan
  - ✓ An airline ticket
  - ✓ A flier announcing a concert
  - ✓ Discount flier at the supermarket
  - ✓ All-you-can-eat ad
  - ✓ Money-back warranty
  - ✓ Contracts!
- Why? Because they make the world more predictable by imposing minimal constraints on actors' behaviour, while respecting their autonomy.

Reasoning with commitments

- Detach: makes a conditional commitment unconditional

  $C(\text{EBook}, \text{Alice}, \$12\text{payment}, \text{BNW}) \wedge \$12\text{payment} \Longrightarrow$

  $C(\text{EBook}, \text{Alice}, \top, \text{BNW})$

- Discharge: $u \Longrightarrow \neg C(x, y, r, u)$
- For example,

  $\text{BNW} \Longrightarrow \neg C(\text{EBook}, \text{Alice}, \$12\text{payment}, \text{BNW})$

  $\text{BNW} \Longrightarrow \neg C(\text{EBook}, \text{Alice}, \top, \text{BNW})$

Operations on commitments

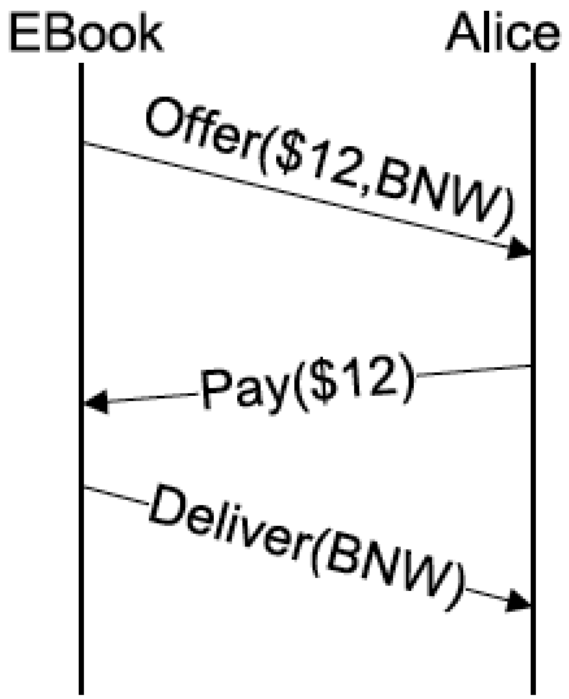
- CREATE(C(x, y, r, u)): performer x, effect C(x, y, r, u)
- CANCEL(C(x, y, r, u)): performer x, effect ¬C(x, y, r, u)
- RELEASE(C(x, y, r, u)): performer y, effect ¬C(x, y, r, u)
- ASSIGN(C(x, y, r, u), z): performer x, effect
  ¬C(x, y, r, u) ⋀ C(x, z, r, u)
- DELEGATE(C(x, y, r, u), z): performer x, effect
  ¬C(x, y, r, u) ⋀ C(z, y, r, u)
- For example,

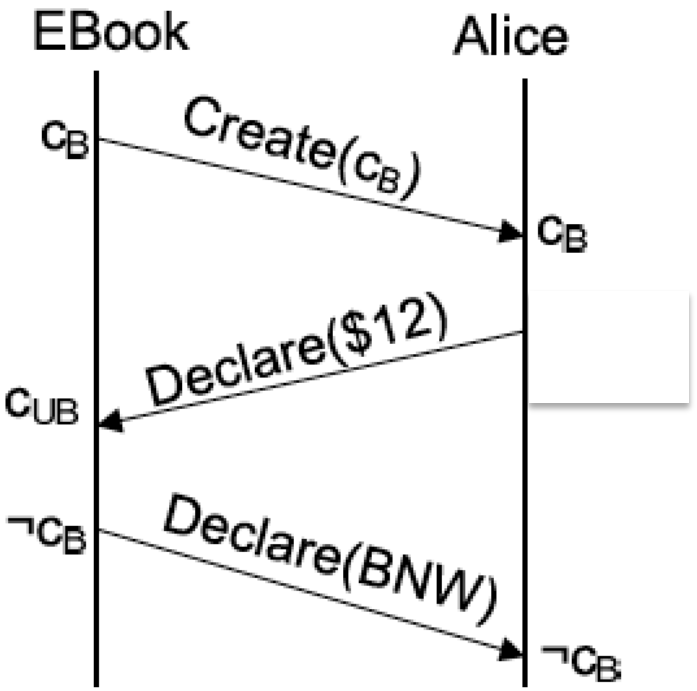
  DELEGATE**(**C**(**EBook**,**Alice**, $**12payment**,**BNW**),**Charlie**)** ⇒
  C**(**Charlie**,**Alice**, $**12payment**,**BNW**)**

  ASSIGN**(**C**(**EBook**,** Alice**, $**12payment**,**BNW**),**Bob**)** ⇒
  C**(**EBook**,**Bob**, $**12payment**,**BNW**)**

Commitments come and go via events

- C_B := C(EBook, Alice, $12payment, BNW))
- C_{UB} := C(EBook, Alice, $\top$, BNW))

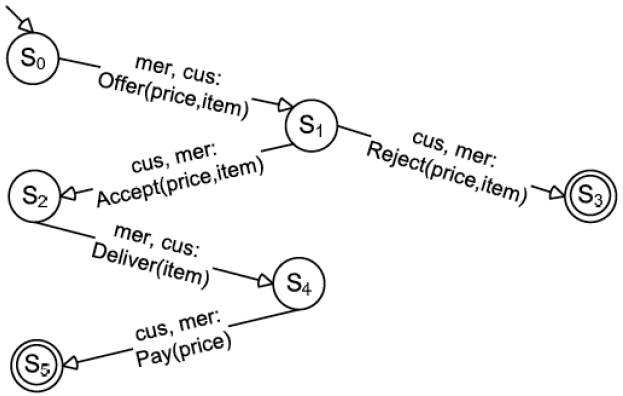




Events ***Meaning***

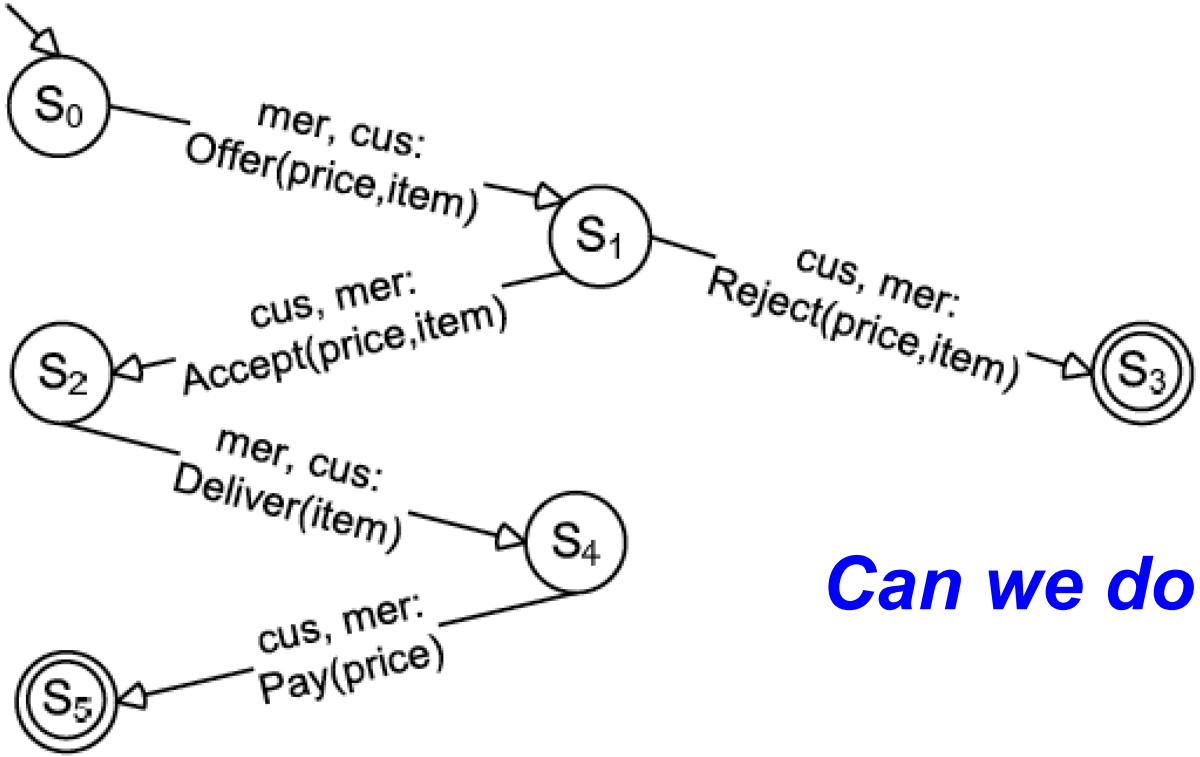
Protocols

- A protocol consists of a sequence of events that create/ discharge commitments among roles.
- For example, a buyer-seller protocol creates a 'sell' commitment upon an offer, which becomes unconditional when a payment is made, and then is discharged when the book is delivered.
- Protocols have been traditionally modelled as state machines/Petri nets, etc.



Challenge: flexible protocols/processes

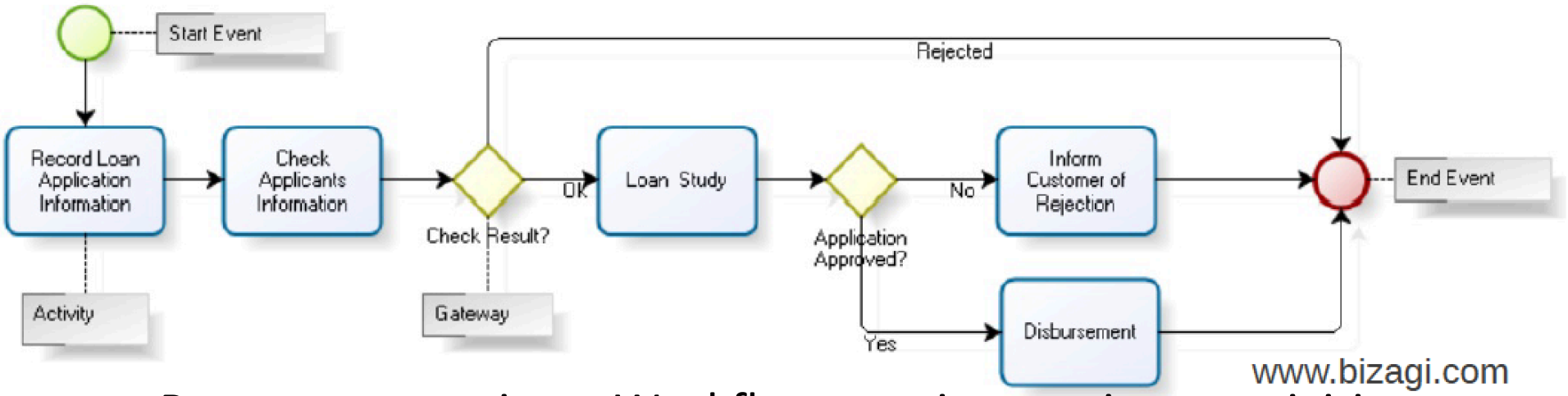
- It is well-known that Petri net/state machine models were intended for ***system*** processes not ***business*** processes as they are not flexible enough and are often violated.



Can we do better?

How to model business processes?

- Right now they modelled as activities and control flows, see BPMN.



- Process execution: Workflow engine assigns activities to performers
- Exception handling by changing assignments, follow alternative flow paths.

Can we do better?

A specification language for business processes (Azzurra)

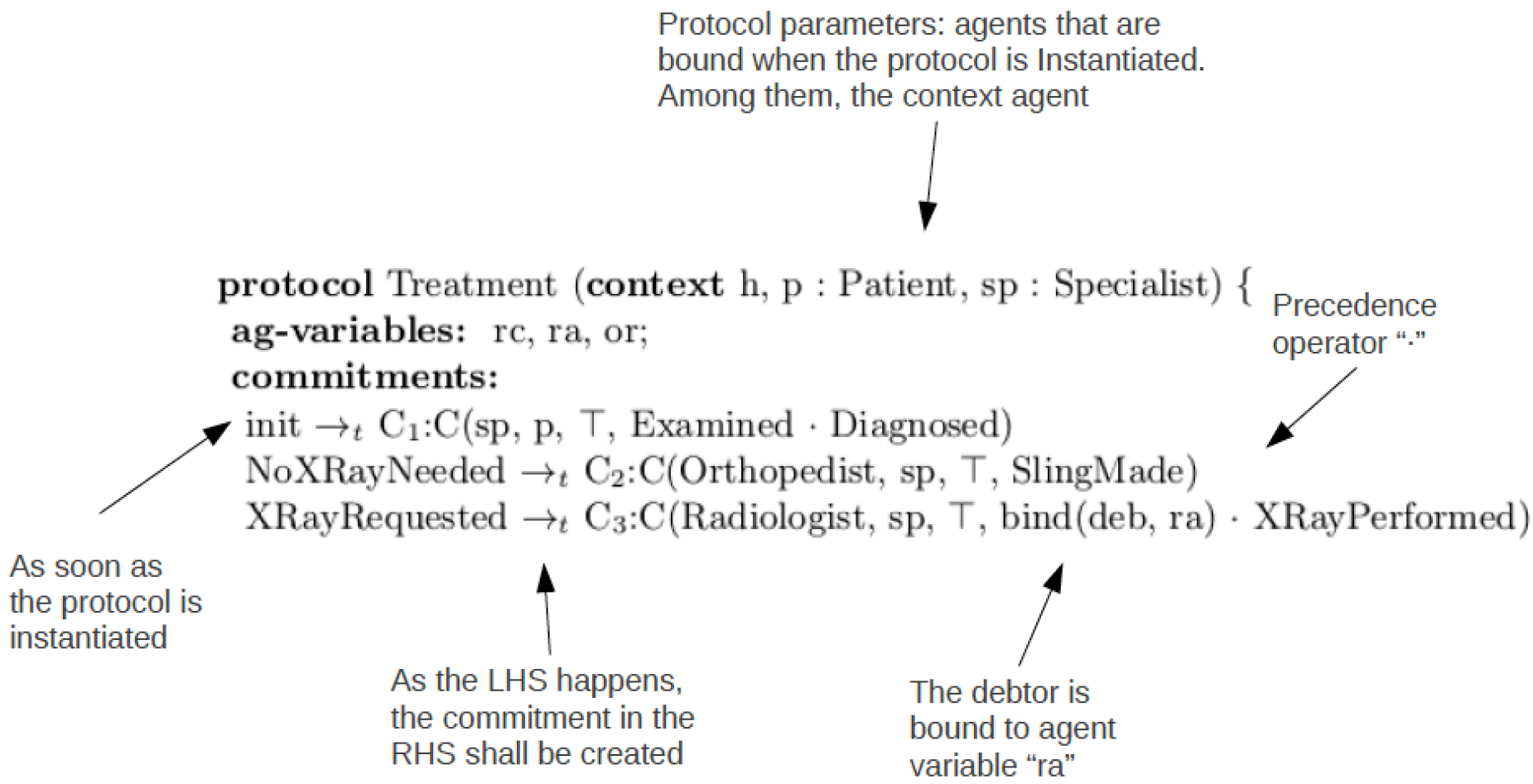
- A business process is specified as a collection of commitments [Dalpiaz15].
- Consider doctor Maria and her commitments:
  - ✓ C1 to head clinician: “Make the daily rounds”
  - ✓ C2 to nurse Zena: “Visit patient Tom by noon”
  - ✓ C3 to secretary Jane: “Fill weekly report by 4pm”
- Here, there are no activities to be performed only outcomes that have to become true, because that’s what commitments are all about!

Azzurra concepts

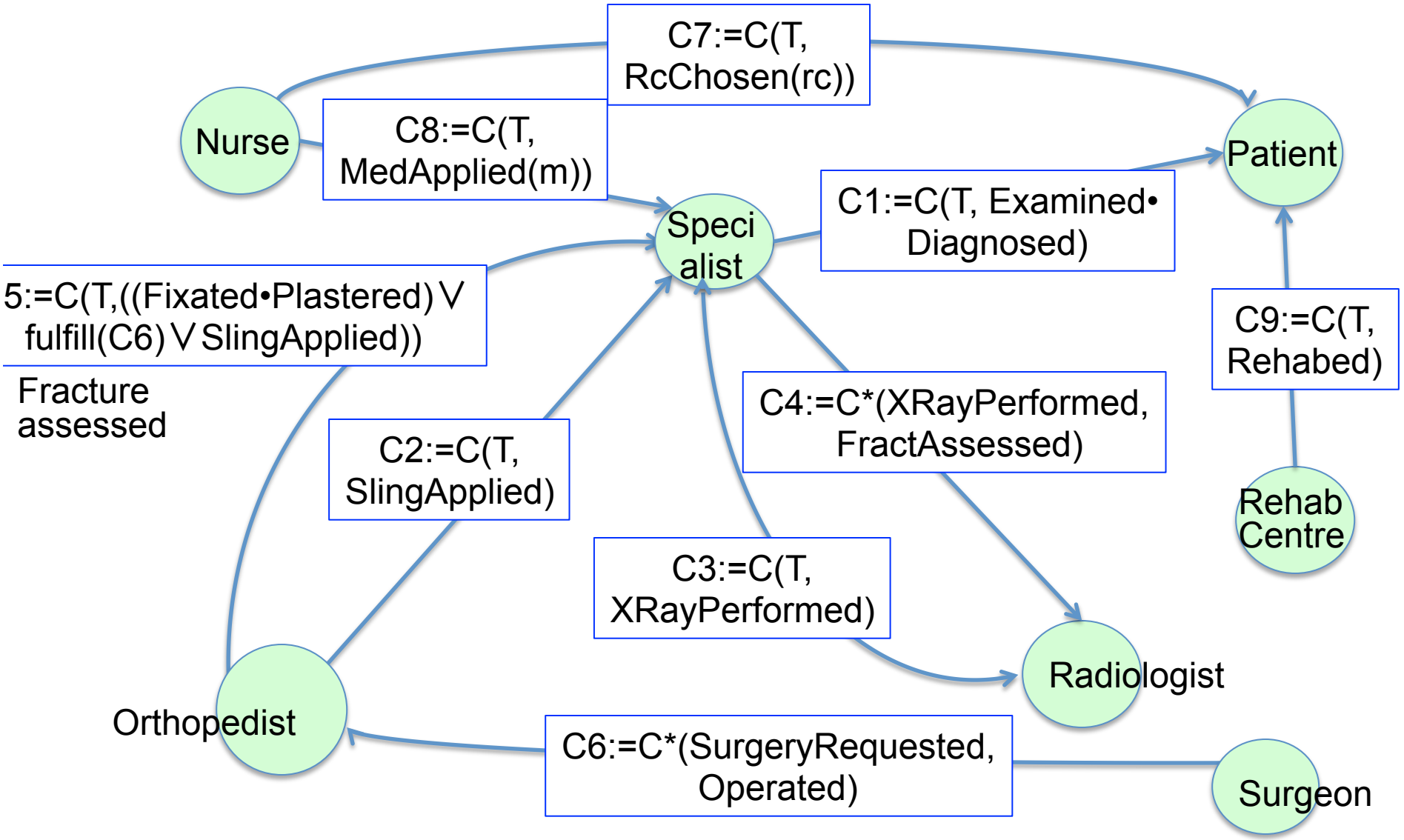
- Agents: Paolo, Vitor
- Roles: Professor, PhDStudent, Postdoc
- Commitment C(x,y,r,u): promise by debtor x to creditor y to make consequent u true if antecedent r becomes true.
- Strong commitment C*(x,y,r,u): promise by debtor x to creditor y to make consequent u true ***after*** antecedent r becomes true.
- Conditional commitment is a commitment that becomes true when a condition holds COND → C(x,y,r,u)
- A protocol declares commitments between roles, as in

C(SeminarOrg, Presenter, TalkDetailsSent, TalkAnnounced)

Example: Fracture treatment protocol



Graphical syntax by example



Reasoning with Azzurra specifications

- Given a sequence of input events such as

  instantiated(Treatment(RGH, John, Maria)),

  instantiated(C1(⋯), Examined(Maria, John),

  Diagnosed(John), noXRayNeeded, ⋯

determine if this is compliant with the specification, and if so, what is the final state of the protocol execution.

Smart contracts

• These are software IoT systems that monitor executions of a contract to ensure that terms are complied with, detect violations and assign blame.

• You can think of them as legal processes, where contract participants are legally bound to do as the contract says.

• Example: Importing meat from Brazil, a contract between a Canadian supermarket chain, a Brazilian meat producer, trucking companies in Brazil and Canada respectively, and a shipping company.

• Here the focus of smart contract software is on ***compliance*** and ***control***.

Smart contract for meat importing

- Receives event logs when meat has been transferred from the producer to the truck company (Brazil), truck company to shipping company etc.
- Receives log when the quality of the meat has been checked, e.g., “should be AAA”, either through sensors or by relying on a human agent.
- Receives logs sent by temperature sensors to ensure that the meat remains refrigerated throughout the trip, except for <2hr periods.
- Checks that no parts of the shipment are substituted with inferior or harmful alternatives.

Smart contracts and blockchains

- For a smart contract to have legal status, the integrity of its data must be guaranteed.
- That's where blockchain technology comes in! This technology ensures that the logs of events recorded in the blockchain ledger (and checked for compliance by smart contract) can't be tampered with.
- Many software companies are selling smart contract systems to supermarket chains, banks, governments, etc.
- What they are actually selling is IoT programs implemented on a blockchain platform.

(SE) Research questions

- Traditionally, software requirements were about ***functions*** and ***quality constraints*** for a system-to-be; but smart contracts are about ***obligations*** and ***powers*** on social agents (aka ***parties)*** and constraints thereof.
- A (formal) specification for a smart contract consists of bilateral obligations and powers with constraints, very different from a specification for vanilla software.
- How do we generate semi-automatically blockchain code from formal specifications?
- Analyzing smart contract specifications using model checkers, SMT solvers, etc.
- How to generate semi-automatically smart contract specifications from natural language?

Symbolaio: Specifications for contracts

- Defined a formal contract specification language
  - ✓ Primitive concepts include ***obligation***, ***power, role, time point/interval***;
  - ✓ Lifetime semantics for obligations, powers;
  - ✓ Limited expressively to Propositional Logic+;
- Working on an analyzer for contract specifications that takes as input a tagged sequence of events

  e1, e2, ⋯, en compliant

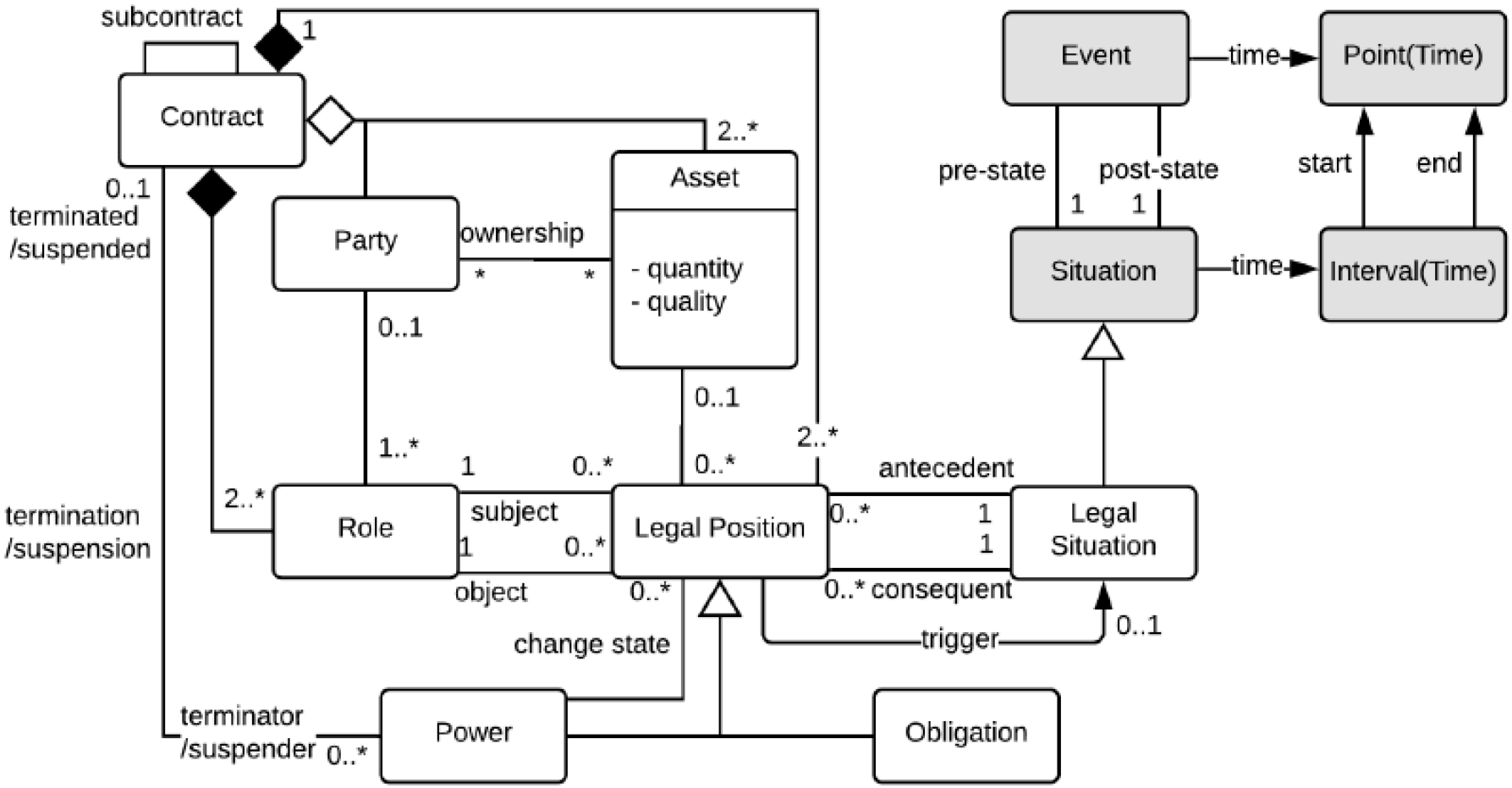
  e1', e2', ⋯, em' non-compliant

and returns CORRECT/INCORRECT, depending on whether the tag is correct or not.

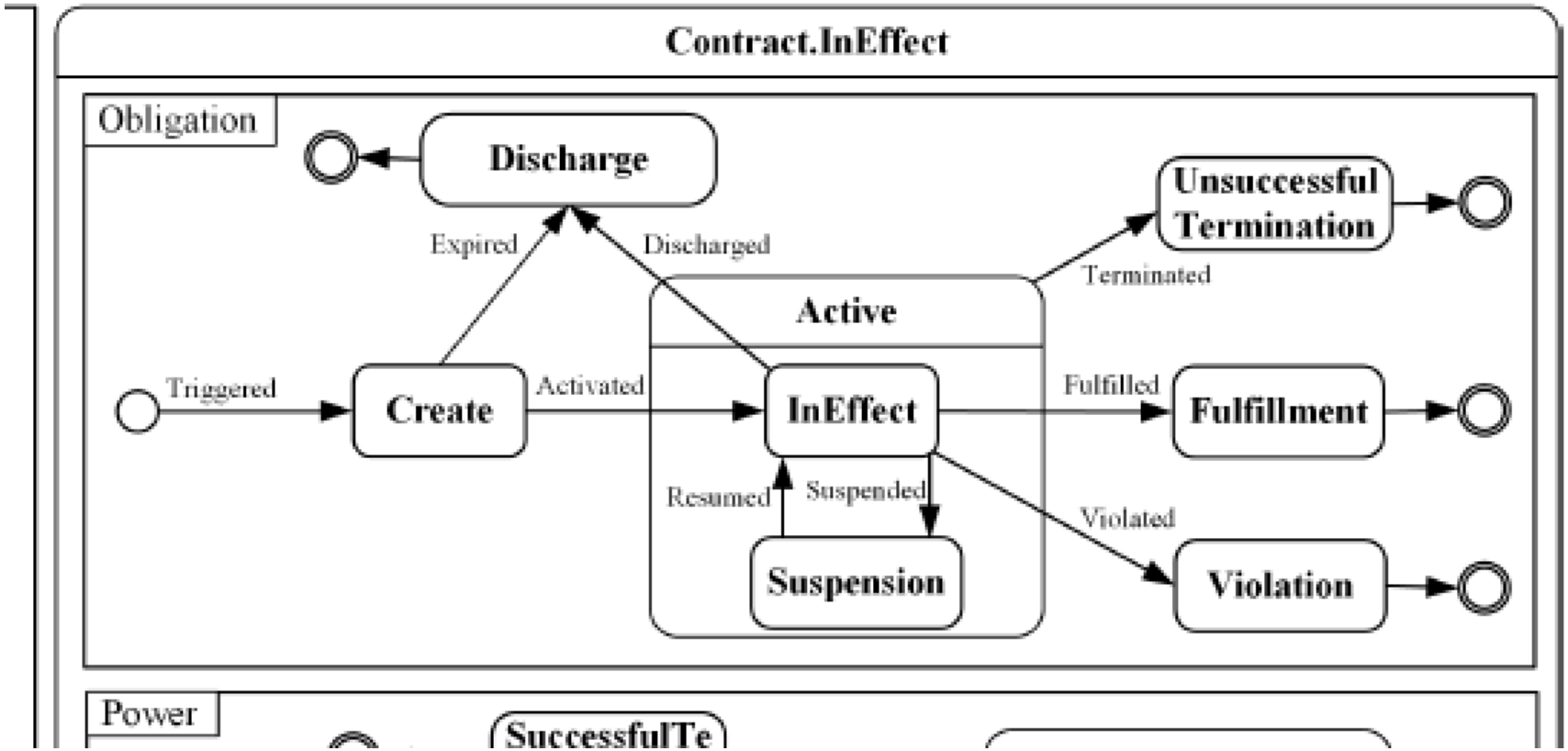
Interesting features

- ***Powers***: these are rights that parties have to change other obligations and powers.
- For example,
  - ✓ "Seller has the power to charge 1.10% the sales price if Buyer is late in paying"
  - ✓ "Money back warranty": "Buyer has the right to get money back if not happy with the purchased goods"
- ***Surviving obligations*** continue to apply after contract execution terminates.
- ***Sub-contracting***, can be done at contract execution time.

Ontology for contracts

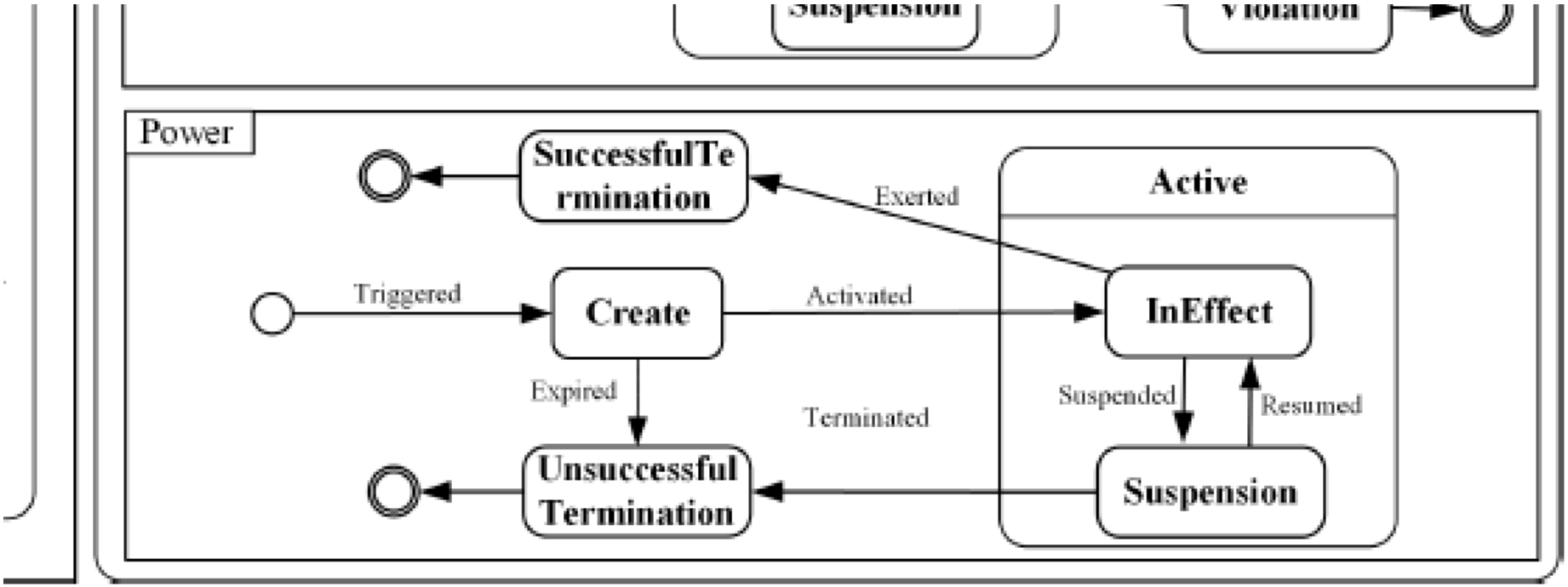


Semantics for obligations



- Semantics consists of defining axioms that specify when do transition events occur; for example "'activated' occurs when the antecedent of an obligation becomes true"

Semantics for powers



Meat purchase and sale agreement

Between Seller and Buyer: This agreement is entered into as of the date **<eDate>**, between **<party1>** as Seller with the address **<retAdd>**, and **<party2>** as Buyer with the address **<delAdd>**.

Terms and Conditions

1. **Payment & Delivery:** (a) Seller shall sell an amount of **<qnt>** meat with **<qlt>** quality (“goods") to the Buyer; (b) Title in the goods shall not pass on to the Buyer until payment of the price has been made in full; (c o The Seller shall deliver the order in one delivery within **<delDueDateDays>** days to the Buyer at its warehouse; (d) The Buyer shall pay **<amt>** (\price") in **<curr>** to the Seller before **<payDueDate>**; (e) In the event of late payment of price due, the Buyer shall pay interests equal to **<intRate>**% of the price.

2. **Assignment:** The rights and obligations are not assignable by Buyer.

3. **Termination:** (a) Any delay in delivery of the goods will entitle the Buyer to terminate the Contract if it exceeds 10 Business Days.

4. **Confidentiality:** (a) Both Seller and Buyer must keep the contents of this contract confidential during the execution of the contract and for six months after the termination of the contract.

Contract domain

- ✓ Seller ***isA*** Role ***with*** returnAddress : String;
- ✓ Buyer ***isA*** Role ***with*** warehouse : String;
- ✓ Meat ***isA*** PerishableGood isA Asset ***with*** quantity : Integer,
  quality : MeatQuality;
- ✓ Delivered ***isA*** Event ***with*** item: Meat, deliveryAddress :
  String, delDueDate : Date;
- ✓ Paid ***isA*** Event ***with*** amount : Integer, currency : Currency,
  from : Buyer, to: Seller, payDueDate: Date;
- ✓ Currency ***isA*** Enumeration(`CAD', `USD', `EUR');
- ✓ MeatQuality ***isA*** Enumeration(`PRIME', `AAA', `AA', `A');
- ✓ …

Contract declarations and conditions

Contract meatSaleC(buyer: Buyer, seller: Seller, qnt: Int, qlt: MeatQuality, amt: Int, cur: Currency, payDueDate: Date, delAdd: String, eDate: Date, del#Days: Int, intRate : Int)

Declarations

goods: Meat ***with*** quantity := qnt, quality := qlt;

delivered: Delivered ***with*** item : goods, deliveryAddress := delAdd, delDueDate := eDate + delDueDatedays;

paid : Paid with amount := amt, currency := curr, from := buyer, to := seller, dueD := payDueDate;

paidLate: Paid with amount := (1 + intRate)*amt, currency := curr, from := buyer, to := seller;

Precondition isOwner(goods, seller);

Postcondition

isOwner(goods, buyer) AND not(isOwner(goods, seller));

Contract obligations

Obligations

o1 : O(seller, buyer, true, happenBefore(delivered,
delivered.delDueDate));

o2 : O(buyer, seller, true, happensBefore(paid,
paid.payDueDate));

o3 : violates(o2) → O(buyer, seller, true,
happens(paidLate,));

SurvivingObligations

so1: SO(seller, buyer, true, keepConfidential);

so2: SO(buyer, seller, true, keepConfidential);

- keepConfidential is defined as “There is no third party that knows the terms of this contract during the execution of the contract and 6mo after”.

Contract powers

Powers

p1: P(seller, buyer, violates(o2 OR o3), suspends(o1));
p2: P(buyer, seller, happensWithin(paidLate, suspension(o1)), resumes(o1));
p3: P(buyer, seller, not(happensBefore(delivered(goods), delivered.delDueDate + 10days), discharges(o2 AND o3) AND terminates(meatSaleC));

Constraints

not(buyer = seller);
∀ o(CannotBeAssigned(o: Obligation);
∀ p(CannotBeAssigned(p: Power);

- Powers are rights that parties have to change another obligation/power, i.e., cancel it, suspend it, etc.

Summary

- Social dependency models are useful not only for specifying requirements assignments in RE, but also for specifying business and legal processes.
- ***Commitments*** constitute a better-developed and formalized variant of *i** ***actor dependencies***.
- Legal ***Obligations/Rights*** are variants of commitments that are legally binding, and therefore have consequences if not complied with.
- The notion of ***power*** is perhaps the most interesting variant among social dependencies in that it allows for a legal process to change completely, depending on what happens with the compliance of initial obligations.

References

- [Bryl09] Bryl V., Giorgini P., Mylopoulos J., "Designing Socio-Technical Systems: From Stakeholder Goals to Social Networks", *Requirements Engineering Journal 14(1)*, 47-70, Springer, 2009.
- [Dalpiaz15] Dalpiaz F., Cardoso E., Cannobio J., GiorginiFa P., Mylopoulos J., "Social Specifications of Business Processes with Azzurra", 9th IEEE International Conference on Research Challenges in Information Science (RCIS'15), Athens, May 2015.
- [Hohfeld13] Hohfeld W., "Some Fundamental Legal Conceptions as Applied in Judicial Reasoning", Yale Law Journal 23(16), 1913.
- [LPG] LPG Homepage. LPG-td Planner. http://zeus.ing.unibs.it/lpg/
- [Sharifi19] Sharifi S., Parvisi-Mosaed A-R., Amyot D., Logrippo L., Mylopoulos J., "Symbolaio: Contracts as Legal Processes with Powers", submitted for publication.
- [Singh91] Singh M., "Social and Psychological Commitments in Multi-Agent Systems", AAAI Fall Symposium, 1991.
- [Singh99] Singh M., "An Ontology for Commitments in Multi-Agent Systems", Artificial Intelligence and Law, 97-113, Kluwer, 1999.
- [Yu95] Yu E., *Modelling Strategic Relationships for Process Reengineering*, PhD Thesis, University of Toronto, 1995.