Foundations of Institutions

Marina De Vos and Julian Padget

Department of Computer Science
University of Bath, England

May 2008

Learning Objectives

- To be able to identify institutions by encounter and to discern the norms that constitute them
- To be able to argue about and justify the classification of individual norms and the relationship between pairs of norms
- To understand a formalization of institutions as a transition system predicated on endogenous and exogenous events
- To understand how such a model may be employed to validate the institutional design and to support the engagement of normative and non-normative agents

The Various Parts

1. Part I: Outline
2. Part II: Agent Societies
3. Part III: Formalising Institutions
4. Part IV: Reasoning about Institutions
5. Part V: Modelling Behaviour
6. Part VI: Programming Institutions

Part II

Agent Societies

Outline

Context

- From Game Theory to Institutions:
    - GT enables strategic analysis
    - ... but games are (relatively) simple
    - participants make bounded rational choices
- Negotiation and Contract Net:
    - Typically one-shot encounters
    - Components in more complex scenarios
- More complex frameworks with stronger guarantees:
    - Coalition: A group of agents, different skills
    - Virtual Organization: A group of agents, subject to an agreed regulatory framework
    - Virtual Institution: A pattern of actions, sanctions, roles and goals

Why Institutions are Essential

- Unconstrained behaviour is not freedom
- (Multiple) Institutions abstract the interaction frameworks needed for constraining behaviour
- Agents can negotiate institutional change
- Institutions can be repositories of emergent behaviour
- Institutions can be formalized and reasoned about with limited computational resources

Is Price Everything?

- Market extreme:
  - Everything has a price
  - $\exists f : A_1 \times ... \times A_n \rightarrow \mathbb{N}$ that is $f(a_1, ..., a_n) \rightarrow p$
  - Hence, the only necessary mechanism is the auction
  - ... and it can be analyzed
- Social extreme:
  - Multi-attribute decision making
  - Social and environmental factors
  - Variety of complex mechanisms
  - ... but it can only be simulated

Analysts vs. Empiricists

- Institutions unite these extremes
  - Enable analytical *and* empirical approaches

Requirements Engineering

- System evolution: closed → semi-open → open
- (Good) governance: evaluating risks and monitoring compliance
- How can component actions be regulated without compromising their integrity or revealing information?
  - Contracts: service level agreements
  - Monitoring/Auditing framework
  - Roles, powers, permissions, authentication
- Virtual ↔ physical world interaction: *counts-as*
- Institutions are a non-invasive way to constrain software components in open architectures

Common Goods

- Common goods: “The Tragedy of the Commons” [2] — an important class of goods that conventional markets cannot (?) handle
  - A resource is shared
  - None has an incentive to restrict their consumption
  - Yet over-consumption will exhaust the resource
  - Examples: water, pasture, fish, bandwidth

A generic problem without a generic solution. For a detailed set of case studies see “Governing the Commons” by Elinor Ostrom [6]

Getting Organized 1/2

- Where’s the organization in self-organization?
  - Typically agents are pre-disposed to coordinate: benevolent
  - Consider sensor networks: relatively rigid frameworks, parameter changes through negotiation
  - Coalitions formed, task completes, cycle repeats: episodic
  - Is this emergent behaviour?
- Can emergence be characterized?
  - Emergence $\stackrel{?}{\Rightarrow}$ interaction + interpretation
  - Consider Steels’ (and others) language games
  - More flexible, but pre-programmed to converge?

Getting Organized 2/2

- How is the interpretation of an interaction defined?
  - Hard-coded: pre-programmed organization
  - Parameterized: adaptable, pre-programmed set of organizations
  - Essentially means agent society is closed
- Furthermore:
  - Which entities may observe the agent’s (inter)actions?
  - How do they observe it?
  - And what does it mean to them?

Institutions: Quality without a Name

- Christopher Alexander [1]: architecture — the design of habitable spaces
- Effective social institutions — social interaction spaces are no different from habitable spaces
- In "Social Laws" [8] Shoham and Tenenholtz identify a more limited objective:

  > *Laws which guarantee the successful co-existence of multiple programs and programmers"*

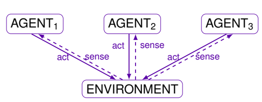
- Task-oriented domains [7]: achievement *vs.* maintenance tasks ≡ "good" final states arising from institutional actions

What is an Agent?

- An agent is a computer system capable of autonomous action in some environment: the situated agent.

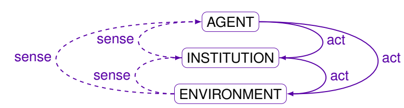
What are *Multi*-Agent Systems?

- An agent can be more useful in the context of others:
  - Can concentrate on tasks within competence
  - Can delegate other tasks
  - Can use ability to communicate, coordinate, negotiate
- So, a MAS is a collection of interacting agents? No:
  - Needs meaningful ways for agents to interact
  - Needs organizational framework
  - Needs identification of roles, responsibilities, permissions
  - Needs to be verified and validated



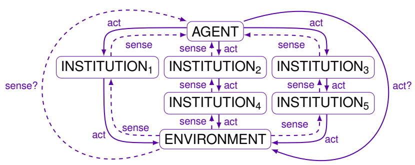
What is an Institution?

- A set of rules:
  - capable of describing correct
  - and incorrect action,
  - obligations acquired through correct action
  - and sanctions levied for incorrect action
  - while maintaining a record through its internal state.
- An institution is a set of rules that interprets some but not necessarily all of an agent's actions as correct or incorrect within that context: the norm-regulated agent.



What is a *Multi*-institution?

- But there is not just one institution
- An agent acts in several institutions, concurrently, even simultaneously
- An institution has restricted competence; aggregation provides complex legal and/or social contexts
- Thus: a multi-institution is a combination of institutions providing the complete interpretation of an agent's actions.



Multiple Institutions

- A single institution can capture the full normative behaviour, but a monolithic structure may be undesirable:
  - Single institutions with a limited range of interaction can be analysed and re-used more easily — institution libraries
  - Institutions are situated in a social and legal framework with whose norms they must interoperate, so institutional workflows are unavoidable
- Institutional composition is a different process in which a *single* internally consistent institution is synthesized from several institutional specifications.
- A multi-institution is a workflow of several connected institutions, each with their own identity and probably with conflicting norms.

Economic motivation

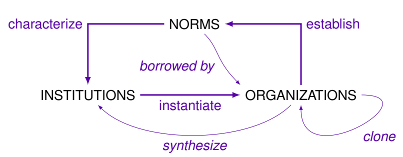
- Douglass C. North in “Institutional Change and Economic Performance” [5] defines:
  - norms that guide and regulate behaviour
  - scenes within which (software) agents may play
  - rôles, while interacting one with another

Social motivation

- A similar point of view comes from the social sciences: Harré and Secord in “The Explanation of Social Behaviour”, [3] define:
  - role-rule model for agent behaviour
  - power being ascribed to agents under a set of conditions
  - episodes in which agents interact and a
  - dramaturgical model that collects + organizes episodes

Institutions *and Norms*

- Assertion: an institution *is* its norms
- What is a norm? Informal or formal constraint on action
- **Definition:** *a principle of right action binding upon the members of a group and serving to guide, control, or regulate proper and acceptable behavior [Merriam-Webster dictionary]*



Institutions

Human institutions have a long history: origins in society or laws made by society.

- So common that we operate unaware of them
- Furthermore we play (or combine a set of) rôles
- Institutions offer a basis for trust and security:
  - decrease uncertainty
  - reduce conflict of meaning
  - create expectations of outcome
  - simplify the decision process

Exercise: The Family

Example

Groups: 2–3 people
Objective: to specify some norms (at various levels) governing the institution of the family.

- What does the family help achieve?
  - Establish objectives
  - Establish values
  - Establish context
- Identify rôles, consider internal and external
- Define two kinds of norm
  - Define an abstract norm
  - Define a concrete norm: consider some or all of rôles, situation, time, concrete terms and actions

Institutions: examples

Institutions are everywhere—formal and informal, legal and social—you just have to know for what to look...

- Conversation, negotiation, argument
- Lecture, seminar, problem class
- Shop: served *vs.* self-service
- Business: sole-trader, partnership, Ltd. company, plc, cooperative, charity, non-profit organization, ...
- Market: stock market, energy trading, brokering (stocks, flights), auction

Norms: examples 1/4

- The distribution of radio frequencies between bidders should take into account their established interests

  Case from the Netherlands in 2002 (?): the government decided to auction off the FM radio band. However, many existing radio stations had their frequency built in to their identity so keeping their existing frequency was more valuable to them.

- If estimated fish stock is x tonnes and viable mass is y tonnes $\Rightarrow$ catch should be $< (x - y)/\#$ fishermen

  A simplistic presentation of fisheries policy that expresses the idea that the total catch should be less than the difference between the total stock and the viable stock—assuming x is greater then y initially.

Norms: examples 2/4

- A front-office trader should not carry out settlements

  In 1995, Nick Leeson's derivatives trading losses of approximately USD1.3b led to the collapse of Barings bank. The mistake: Leeson as Chief Trader was also permitted to settle his accounts—rôles that are normally separated—and hence hide his losses for many months.

  Latterly the Enron collapse led to the Sarbanes-Oxley Act (SOX) which requires the maintenance of specific audit trails and the separation of specified responsibilities. See `http://en.wikipedia.org/wiki/Sarbanes-Oxley_Act`.

Norms: examples 3/4

- Polluter pays? Kyoto protocol, carbon credits

  A new kind of currency has been created with the Kyoto protocol, which enables countries and soon maybe individuals to trade in pollution.

- Who should have this liver?

  Conventional organ distribution mechanisms are somewhat arbitrary, depending as much on personal acquaintance and chance as medical compatibility. Communication is often haphazard, involving telephone and fax. With more sophisticated distribution mechanisms (see Carrel [9]) policies can be applied, such as old-for-the-old, use of organs from hepatitus-C donors etc.

Norms: examples 4/4

- Don't led market players design your mechanism

  Why did California black-out? Because it couldn't pay its bills. Why couldn't it pay its bills? Because prices were artificially inflated. Why were prices high? Because one market player knew how to manipulate the market. Who was that player? Enron. Who designed the market? Enron.

Degrees of precision and enforceability emerge...

The purpose of these examples and the two case studies (next) is to make the case that simple markets in isolation are inadequate in complex situations.

Categorization of norms

- Constitutive, also known as abstract norms: high level, expressing what *ought* (not) to be
- Regulatory norms: middle level, capturing actions or changes of state that depend on particular conditions
- Procedural norms, also called protocols: sequences of actions that (typically) if followed ensure (higher level) norm-compliance
- The technical challenge is how to prove that a specification at one level is consistent with one at another level

Pre-privatisation

- Originally, electricity generation was controlled by a single (nationalised) organization, the Central Electricity Generating Board (CEGB) which owned and controlled the power stations and the means of distribution (the national grid). Domestic distribution controlled by local monopolies.
- Privatisation split the CEGB up into 4 power generating companies (PowerGen, Nuclear Electric,...) and one distribution company (National Grid Company). There were also several smaller generators. The NGC bought electricity and distributed to the domestic distribution companies.

The NGC and the “Pool”

- The pool operated on a 24hr basis, where each generator offered to supply *x* Megawatts for the period. NGC ranked the bids by price (from lowest to highest) and at 17:00 every day, it accepted as many of those bids as needed to meet predicted energy requirements. All accepted bids were paid at the price of the highest accepted bid.
- Consequently, the pool could be manipulated by the largest generators who could predict the bid cut-off and thus put forward over-priced bids that were ranked around the predicted national energy requirement.

New Electricity Trading Arrangements (NETA)

A four phase market (see `www.ofgem.gov.uk`) started March 27th 2001:

- **network access rights:** auction bundles of tickets (1MegaWatt (MW) per half-hour period) at regular intervals
- **power/access trading:** unfacilitated bi-lateral generator / distributor trades (inform system operator (SO)) of 0.5hr periods from 6 months to 3.5hrs before dispatch (gate closure). Access trading unimplemented. Emergence of power exchanges.
- **balancing market:** 3.5hr period prior to dispatch, controlled by the System Operator (National Grid Company (NGC)). Objectives: prevent thermal overload; prevent dynamic instability. Actions: back-off generation; shed load, increase generation; absorb excess.
- **settlement:** (`www.elexon.co.uk`) distributors pay generators, NGC pays additional generation, compensates backed-off generation, distributors/generators pay penalties for being short/long.

NETA: analysis

- Secondary market enables forward and futures trading, also permits entry to market of traders without a physical position, e.g. Enron.
- Transmission costs are not integrated—too complicated for timescale; may be too unpredictable?
- Oriented to large-scale power trading ($>$5MW), inhibiting the participation of co-generation facilities and green power sources.
- Under the pool system the nuclear generators always bid low, because switching nuclear plant on and off is a time-consuming and costly process. But, because of the pool, they were remunerated at a high level. Under NETA this was not possible.
- *Significant* modification needed to accomodate trading and generation at the domestic level.
- High penalties for being “short” or “long”, i.e. not having tickets to match generated output.

NETA: outcomes

- Achieved primary objectives:
  - Wholesale price of electricity reduced
  - Price manipulation less feasible
- Unintended consequences:
  - Nuclear generation capacity bankrupted—because of reduction in wholesale price
  - Green generation rendered non-viable—because of penalties

BETTA

British Electricity Trading and Transmission Arrangements:

- Go-live April 1, 2005
- Introduced transmission charging
- The Transmission Network Revenue Restriction (placed on the National Grid Company) states that:

  The licensee shall use its best endeavours to ensure that in any relevant year the revenue from its transmission network services shall not exceed the maximum revenue, which shall be calculated in accordance with the following formula:

$$M_t = \left[I + \frac{RPI_t - X_g}{100}\right]P_{t-1} - D_t - K_t + G_t + CCC_t + LPC_t + LPR_t + RI_t + IES_t + TSP_t + TSH_t$$

- More details at `www.ofgem.gov.uk`

The (ab)use of money 1/2

- What does money do?
  - Metric for value comparison
  - Efficient medium of exchange
- But soon new uses arose:
  - Store of value
  - Tool for speculation

In 1980s 40% of currency trading was to support international trade. By the end of 1990s had fallen to 2%, while volume had increased significantly.

The (ab)use of money 2/2

- Volume of hoarding and speculation dominate creating
  - volatility
  - and instability

  in financial (and commodity) markets with significant consequences for economies and local markets.

  > *"Instability is cumulative, so that eventual breakdown of freely floating exchanges is virtually assured.", George Soros.*

- Can anything be done?
- Use complementary currencies: encourages local trading; necessity when liquidity is low — because liquidity is created at the point of trade.
- A case of mechanism design...

A matrix of currency systems

	scarce, competition-promoting	**sufficient, cooperation-promoting**
"Fiat": authority guaranteed	Today's national currencies	Ithaca HOURS
"Backed": external reference guaranteed	• frequent flyer miles • barter • e-gold • Global Reference Currency	Mutual credit systems: • time dollars • LETS • ROCS

Use the right currency (mechanism) for the kind of transaction: economically rational but politically problematic

Characteristics of CC models 1/2

- Benefits:
  - no need for central authority to guarantee or manage (backed currencies)
  - mutual credit systems (no shortage of currency) encourage cooperation
  - generates more transactions leading to greater satisfaction and creating more trading relationships
- Issuing currency: "fiat" *or* mutual credit *or* commodity-backed — redeemable for good or service
- Assigning value: hour of service *or* direct correspondence with a fiat currency *or* goods/services themselves define value

Characteristics of CC models 2/2

- Exchanging rates: fixed (time dollars) — one person's time has the same value as another's *or* negotiable (most others) — diverse communities
- Structural incentives:
  - CCs do not bear interest: encourages investment in production rather than hoarding; encourages trade and cooperation because money supply is plentiful
  - *Demurrage* (Silvio Gesell) — negative interest — may be applied as a disincentive to hoarding... egg currency
  - Backed currencies are less susceptible to inflation than fiat.
- Working systems: Ithaca HOURS; LETS; Time Dollars; e-gold

Note: adapted from `http://www.transaction.net`

Currency for internet institutions

- Recipe book for building a transaction framework
- ROCS – Robust Complementary Currency System (`www.transaction.net/money/rocs/`) describes a complete system
- Complementarity of CCs and Internet:
  - Rôle of internet as facilitator for CCs
  - Rôle of CCs as facilitators of internet trading
- Not just a mechanism for people to trade goods and services, but principles for the design of (financial) instruments whereever valuations and (dis)incentives are needed.
- See "The Future of Money" [4].

References

- Christopher Alexander.
  A Timeless Way of Building.
  Center for Environmental Structure. Oxford University Press Inc, USA, 1980.
  ISBN-13: 978-0195024029.
- Garrett Hardin.
  The Tragedy of the Commons.
  Science, 162:1243–8, 1968.
- R Harré and P.F. Secord.
  The Explanation of Social Behaviour.
  Blackwells, 1972.
  ISBN 0 631 14220 7.
- Bernard Litaer.
  The Future of Money: Creating New Wealth, Work and a Wiser World.
  Century, 2002.
  ISBN: 0712699910. Large extracts available on-line at `www.transaction.net`.
- Douglass C. North.
  Institutions, Institutional Change and Economic Performance.
  Cambridge University Press, 1991.
- Elinor Ostrom.
  Governing the Commons. The Evolutions of Institutions for Collective Action.
  Cambridge University Press, Cambridge., 1990.
- Jeffrey S. Rosenschein and Gilad Zlotkin.
  Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.
  Artificial Intelligence. MIT Press, 1994.
  ISBN-13: 978-0-262-18159-4.
- Yoav Shoham and Moshe Tennenholtz.
  On social laws for artificial agent societies: off-line design.
  Artif. Intell., 73(1-2):231–252, 1995.
- Javier Vázquez-Salceda, Julian Padget, Ulises Cortés, Antonio López-Navidad, and Francisco Caballero.
  Formalizing an electronic institution for the distribution of human tissues.
  Artificial Intelligence in Medicine, 27(3):233–258, 2003.
  ISSN: 0933-3657.

Exercise: Intelligent vehicles

Example

- **Groups:** 2–3 people
- **Objective:** Identify ways in which intelligent vehicles might use ad-hoc networks to find resources or to organize themselves
- **Plan:**
  - Core activity
    - Identify potential scenarios
    - Choose one to explore in more detail
    - Consider what information is needed (sources) and what communication is required
    - Identify expected outcomes
    - Repeat as desired
  - Reflect and discuss

Summary of Organizations, Institutions and Norms

- Proposed a relationship between the concepts of organization, institution and norm
- Identified the situated institution as a form of mechanism design in a spectrum that puts game theory at the opposite end
- Developed the abstract idea of interaction frameworks as a generalization of the economic notion of institution.
- Concept of norm: characterises an institution, captures constraints on behaviour with variable degrees of precision.

Related Work 1/2

- Not exhaustive, not exclusive!
- Esteva et al in [1]
  - Describes the Islander (now called e-Institutor) toolkit for the specification and animation of institutions
  - Visual programming interface–like diagrams earlier
  - Relatively weak (at the time) support for norms
- Vázquez-Salceda [5]
  - Sets out the idea of layers of norms and how they may be related
  - Unifying notion of policy
  - Concrete example from organ transplant

Related Work 2/2

- Dignum [6]
  - Describes the specification of organizations in terms of landmarks
  - Moves away from the protocol-based approach
  - Identifies need for logic-based techniques in proving properties of organizations
- Don't forget there is a rich literature outside computer science, specifically in economics [3, 4] and in social sciences [2] and in law/legal reasoning.

References

Marc Esteva, David de la Cruz, and Carles Sierra.
ISLANDER: an electronic institutions editor.
In *AAMAS*, pages 1045–1052. ACM, 2002.

R Harré and P.F. Secord.
The Explanation of Social Behaviour.
Blackwells, 1972.
ISBN 0 631 14220 7.

Douglass C. North.
Institutions, Institutional Change and Economic Performance.
Cambridge University Press, 1991.

Elinor Ostrom.
Governing the Commons. The Evolutions of Institutions for Collective Action.
Cambridge University Press, Cambridge., 1990.

Javier Vázquez Salceda.
The role of Norms and Electronic Institutions in Multi-Agent Systems applied to complex domains.
PhD thesis, Technical University of Catalonia, 2003.

Virginia Dignum.
A Model for Organizational Interaction Based on Agents, Founded in Logic.
PhD thesis, Utrecht University, 2004.

Part III

Formalising Virtual Institutions

Outline

Characteristics of virtual institutions

Who: actors
- internal
- external

+

What:
- actors may say
- actors may do

When:
- a communication may occur
- an action may take place

Where:
- an actor may go
- an action may take place

+

State:
- records
- obligations

Observable actions of agents change the institution's state

A Norm-driven approach

- A top-down approach to institutional modelling views an institution as:
  - A set of *institutional states* that evolve
  - in response to *institutional events*.
  - where an institutional state is a set of *institutional facts*
- State changes are a result of the *observables* identified earlier

A Norm-driven approach

- How are institutional facts created?
  - Searle [2] identifies two kinds of facts
    - Brute facts that are observable in the physical world
    - and institutional facts that are neither observable, nor have any meaning outside their institution
  - Institutional facts are created by an action in the physical world that counts as taking that action in the institutional world.
  - Thus the observation of an agent action can lead to the creation of an institutional fact within the institution in which the agent is participating.

Social States

Several types of institutional facts are considered:

- **Permission:** An agent's Ability to carry out some action without sanction.
- **Obligation:** Facts stating that an agent is obliged to have done some action before some deadline.
- **Institutional Power:** (after Jones & Sergot) institutional facts describe an agent's capacity to affect the social state by performing *meaningful* institutional actions. (i.e. power to conduct a valid marriage)
- **Domain Facts:** Those relating internally to the institution in question. (i.e. marina Ows X)

Events

- Account for (possible) changes in state
- May be:
  - Domain Events (exogenous): observed from the environment.
  - Institutionally generated (internal): generated by the institution
- Events may generate other events: Conventional generation.

Conventional Generation

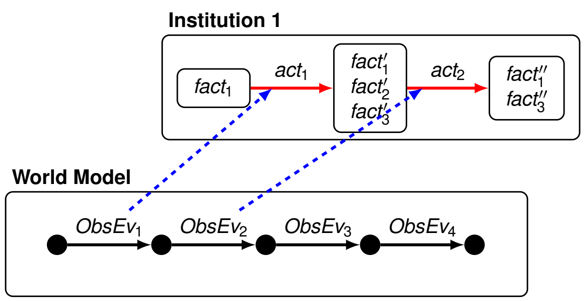
Origins in theory of action (Goldman, Searle, Jones & Sergot)

- "Doing X [in environment A] counts as doing Y [in environment B] iff Z"
- Allows us to abstract institutional actions from real world ones, i.e.:
  - "Saying 'aye' in an auction counts as an offer to buy some goods at the current price"
  - "Clicking 'buy it now' counts as an offer to buy some goods at a given price on amazon"
- Generation is assumed to be atomic (i.e. generated events occur concurrently with events which generate them)

Regulation

- Not all sequences of action are desirable
- We specify regulatory rules to identify "bad" paths of events
- Two regulatory mechanisms are considered:
  - **Obligation:** "You should do X before Y happens"
  - **Permission:** "You should not do X"
- **Violations:** When the above rules are broken *violation events* are generated for:
  - The failure to perform an action before a deadline.
  - Performing an action without permission.

Specification Model



Formal Specification

Definition

Institutions: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{C}, \mathcal{G}, \Delta \rangle$

- $\mathcal{E} = \mathcal{E}_{obs} \cup \mathcal{E}_{inst}$ with $\mathcal{E}_{inst} = \mathcal{E}_{instact} \cup \mathcal{E}_{viol}$
- $\mathcal{F} = \mathcal{W} \cup \mathcal{P} \cup \mathcal{O} \cup \mathcal{D}$
- $\mathcal{C} : \mathcal{X} \times \mathcal{E} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}}$ with $C(X, e) = (\mathcal{C}^{\uparrow}(X, e), \mathcal{C}^{\downarrow}(X, e))$
- $\mathcal{G} : \mathcal{X} \times \mathcal{E} \to 2^{\mathcal{E}_{inst}}$
- Δ
- State Formula: $\mathcal{X} = 2^{\mathcal{F} \cup \neg \mathcal{F}}$

where institutional facts ($\mathcal{F}$) are defined in terms of power ($\mathcal{W}$), permission ($\mathcal{P}$), obligation ($\mathcal{O}$) and domain facts ($\mathcal{D}$) and where $\mathcal{C}^{\uparrow}(X, e)$ and $\mathcal{C}^{\downarrow}(X, e)$ resp., contain those fluents which are *initiated/terminated* by the event e in any state matching X

Semantics

- Event Generation.
- Fluent Initiation.
- Fluent Termination
- State Transformation
- Traces

Event Generation

GR(S, E) generates

Intuition

1. Events that are generated remain generated
2. Empowered Events which are generated from conventional generation with conditions matching S
3. Violations generated from conventional generation matching the current state
4. Violations that result from events which were not permitted
5. Violations from obligations for which the deadline has expired

Event Generation

Definition

$$
\begin{array}{lll}
\mathrm{GR}(S,E) = \{e \in \mathcal{E} \mid & e \in E & \textbf{or} \\
& \exists\, e' \in E, \phi \in \mathcal{X}, e \in G(\phi, e') \cdot S \models \mathrm{pow}(e) \land S \models \phi & \textbf{or} \\
& \exists\, e' \in E, \phi \in \mathcal{X}, e \in G(\phi, e') \cdot e \in \mathcal{E}_{viol} \land S \models \phi & \textbf{or} \\
& \exists\, e' \in E \cdot e = viol(e'), S \models \neg\, \mathrm{perm}(e') & \textbf{or} \\
& \exists\, e' \in \mathcal{E}, d \in E \cdot S \models \mathrm{obl}(e', d, e)\} &
\end{array}
$$

Initiation

Intuition

Fluents are initiated if: If a certain event in the current environment triggers the consequence relation to initiate this fluent

Definition

$$\mathrm{INIT}(S, e_{obs}) = \{p \in \mathcal{F} \mid \exists\, e \in \mathrm{GR}^{\omega}(S, \{e_{obs}\}), X \in \mathcal{X} \cdot p \in \mathcal{C}^{\uparrow}(X, e) \land S \models X\}$$

Termination

Intuition

Fluents are terminated if:

- A certain event in the current environment triggers the consequence relation to terminate this fluent, or
- The deadline or the event of an obligation occurred

Definition

$$
\begin{array}{lll}
\mathrm{TERM}(S, e_{obs}) = \{p \in \mathcal{F} \mid & \exists\, e \in \mathrm{GR}^{\omega}(S, \{e_{obs}\}), X \in \mathcal{X} \cdot p \in \mathcal{C}^{\downarrow}(X, e), S \models X & \textbf{or} \\
& p = \mathrm{obl}(e, d, v) \land p \in S \land e \in \mathrm{GR}^{\omega}(S, \{e_{obs}\}) & \textbf{or} \\
& p = \mathrm{obl}(e, d, v) \land p \in S \land d \in \mathrm{GR}^{\omega}(S, \{e_{obs}\})\} &
\end{array}
$$

State Transitions

Intuition

The new states consists of the fluents of the old state which were not terminated plus all the newly initiated fluents.

Definition

We define the transition function $\mathrm{TR} : \Sigma \times \mathcal{E}_{obs} \rightarrow \Sigma$ as:

$$
\begin{array}{ll}
\mathrm{TR}(S, e_{obs}) = \{p \in \mathcal{F} \mid & p \in S, p \notin \mathrm{TERM}(S, e_{obs}) \quad \textbf{or} \\
& p \in \mathrm{INIT}(S, e_{obs})\}
\end{array}
$$

Traces

- An ordered trace is defined as a sequence of observable events

$$\langle e_0, e_1, \ldots, e_n \rangle \quad e_i \in \mathcal{E}_{obs}, 0 \leq i \leq n$$

- The evaluation of an ordered trace for a given starting state S_0 is a sequence $\langle S_0, S_1, \ldots S_{n+1} \rangle$ such that $S_{i+1} = \mathrm{TR}(S_i, e_i)$
- Ordered traces and their evaluations allow us to monitor or investigate the evolution of an institution over time. They provide us with the data necessary to answer most queries one might have about a certain institution.

An example

Example

A country is constantly swinging between war and peace with its neighbour. The countries have agreed that when they are at peace, a citizen of the first shooting a citizen of the second counts as murder, when they are at war and a citizen has been conscripted into the army it is permitted to shoot. When one country is provoked, it is obliged to start war first before it is allowed to shoot.

An example

Example

$$
\begin{aligned}
\mathcal{E}_{obs} &= \{\texttt{shoot}, \texttt{startwar}, \texttt{declaretruce}, \texttt{callup}, \texttt{provoke}\} && (1)\\
\mathcal{E}_{instact} &= \{\texttt{conscript}, \texttt{murder}\} && (2)\\
\mathcal{E}_{viol} &= \{\mathsf{viol}(\texttt{shoot}), \mathsf{viol}(\texttt{startwar}), \mathsf{viol}(\texttt{declaretruce}),\\
&\quad \mathsf{viol}(\texttt{callup}), \mathsf{viol}(\texttt{provoke}), \mathsf{viol}(\texttt{conscript}), \mathsf{viol}(\texttt{murder})\} && (3)\\
\mathcal{D} &= \{\texttt{atwar}\} && (4)\\
\mathcal{W} &= \{\mathsf{pow}(\texttt{conscript}), \mathsf{pow}(\texttt{murder})\} && (5)\\
\mathcal{P} &= \{\mathsf{perm}(\texttt{shoot}), \mathsf{perm}(\texttt{startwar}), \mathsf{perm}(\texttt{declaretruce}),\\
&\quad \mathsf{perm}(\texttt{callup}), \mathsf{perm}(\texttt{provoke}), \mathsf{perm}(\texttt{conscript}), \mathsf{perm}(\texttt{murder})\} && (6)\\
\mathcal{O} &= \{\mathsf{obl}(\texttt{startwar}, \texttt{shoot}, \texttt{murder})\} && (7)
\end{aligned}
$$

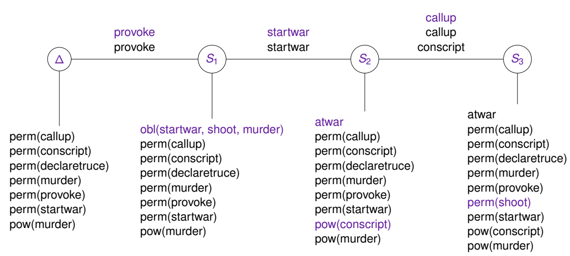
An example

Example

$$
\begin{aligned}
\mathcal{C}^{\uparrow}(\mathcal{X},\mathcal{E}): \quad & \langle\{\neg \mathit{atwar}\}, \texttt{startwar}\rangle && \to \{\mathit{atwar}\} && (8)\\
& \langle\{\neg \mathit{atwar}\}, \texttt{provoke}\rangle && \to \{\mathsf{obl}(\texttt{startwar}, \texttt{shoot}, \texttt{murder})\} && (9)\\
& \langle\emptyset, \texttt{conscript}\rangle && \to \{\mathsf{perm}(\mathit{shoot})\} && (10)\\
& \langle\emptyset, \texttt{startwar}\rangle && \to \{\mathsf{pow}(\mathit{conscript})\} && (11)\\
\mathcal{C}^{\downarrow}(\mathcal{X},\mathcal{E}): \quad & \langle\{\mathit{atwar}\}, \texttt{declaretruce}\rangle && \to \{\mathit{atwar}\} && (12)\\
& \langle\emptyset, \texttt{declaretruce}\rangle && \to \{\mathsf{perm}(\mathit{shoot})\} && (13)\\
& \langle\emptyset, \texttt{declaretruce}\rangle && \to \{\mathsf{pow}(\mathit{conscript})\} && (14)\\
\mathcal{G}(\mathcal{X},\mathcal{E}): \quad & \langle\emptyset, \texttt{callup}\rangle && \to \{\mathit{conscript}\} && (15)\\
& \langle\emptyset, \texttt{viol(shoot)}\rangle && \to \{\mathit{murder}\} && (16)
\end{aligned}
$$

$$
\begin{aligned}
S_0 &= \{\mathsf{perm}(\mathit{callup}), \mathsf{perm}(\mathit{startwar}), \mathsf{perm}(\mathit{conscript}), \mathsf{perm}(\mathit{provoke}),\\
&\quad \mathsf{pow}(\mathit{murder}), \mathsf{perm}(\mathit{murder})\} && (17)
\end{aligned}
$$

Example



An Exercise: Borrowing

Example

The borrowing institution:

- describes when agents may borrow money
- when they have to pay it back
- when they are permitted to leave
- The norm: when money is borrowed it must be payed back before the agent leaves
- Observable events are generated by the environment, not the agents themselves

In pairs, write the formal specification for this institution.

Summary of Single Institutions

- Provided a formalisation of an institution, allowing for empowerment, permission and obligation
- Traces provide the evolution of an institution over time

Reading material

Owen Cliffe, Marina De Vos, and Julian Padget.
Answer set programming for representing and reasoning about virtual institutions.
In Katsumi Inoue, Satoh Ken, and Francesca Toni, editors, *Computational Logic for Multi-Agents (CLIMA VII)*, volume 4371 of *LNAI*, pages 60–79, Hakodate, Japan, May 2006. Springer.

John R. Searle.
The Construction of Social Reality.
Allen Lane, The Penguin Press, 1995.

Part IV

Reasoning about Institutions

Outline

Reasoning in the formal model

- The formal spefications allows us to describe all the components of an institution in a very concise and precise way
- however, it comes with little functionality to validate or reason about the institution
- unless we want to do everything manually
- so we need a computational tools

Computational Tools

- Based on logic to assure verifiability
- Expressive
- Straightforward mapping
- Queries
- We use answer set programming for our modelling
  - Sound grounding in logic - verifiable
  - Specification equals implementation
  - Intuitive
  - Very expressive

Answer Set Programming Paradigm

- Fundamental concept:

  Models, not proofs, represent solutions!
- Therefore, need techniques to compute models (not to compute proofs)

  ASP Solvers
- What is model generation good for?

  Solve search problems
  - Reasoning about ontologies
  - Error diagnoses for a faulty system ;
  - Music Synthesis
  - Evolution of language
  - Optimal code sequences
  - Agent Reasoning

Intuition

- Clauses:

Definition

$\mathtt{a} : -\mathtt{b}_1, \ldots, \mathtt{b}_m, \textbf{not } \mathtt{c}_1, \ldots, \textbf{not } \mathtt{c}_n.$

Intuition

If we believe $\mathtt{b}_i$ and we do not believe $\mathtt{c}_j$ then we have to believe $\mathtt{a}$

- Semantics:

Inituition

We only want to believe those atoms that have the full support of a rule.

A Few Extra Constructs...

Constraints $\perp \leftarrow a, b.$
If we know a and b we don't have an answer set

Predicates $a(X, Y) \leftarrow b(X), c(Y).$
$a(1, 1) \leftarrow b(1), c(1).,\ a(1, 2) \leftarrow b(1), c(2).,$
$a(2, 2) \leftarrow b(2), c(2)., \ldots$

Choice Rules $1\ \{a(1), a(2), a(3), a(4)\}\ 3 \leftarrow b.$
If we know b then we know between 1 *and* 3 *of* $a(1), a(2), a(3), a(4)$

And functions, classical negation, variable domains, preferences, ...

Graph Colouring

Example

$$1\ \{paint(N, C) : colour(C)\}\ 1 \leftarrow node(N).$$
$$\perp :- node(N1), node(N2), N1 \neq N2, link(N1, N2), paint(N1, C), paint(N2, C), colour(C).$$

Per instance add *colour*, *node* and *link*.

Can you guess what has been encoded?

Example

```
position(1 .. 9).
value(1 .. 9).
1 { state(X,Y,N) : value(N) } 1 :- position(X), position(Y).
:- state(XA,Y,N), state(XB,Y,N), position(XA), position(XB),
  position(Y), value(N), XA != XB.
:- state(X,YA,N), state(X,YB,N), position(X), position(YA)
 , position(YB),value(N), YA != YB.
sameSubSquare(1,2).
sameSubSquare(2,3).
sameSubSquare(4,5).
sameSubSquare(5,6).
sameSubSquare(7,8).
sameSubSquare(8,9).
sameSubSquare(X,Y) :- sameSubSquare(Y,X), position(X), position(Y).
sameSubSquare(X,Z) :- sameSubSquare(X,Y), sameSubSquare(Y,Z),
 position(X), position(Y), position(Z).
:- state(XA,YA,N), state(XB,YB,N), sameSubSquare(XA,XB),
  sameSubSquare(YA,YB), position(XA),
  position(XB), position(YA), position(YB),
  value(N), XA != XB, YA != YB.
```


Exercise

Example

Write an ASP program that given a set of facts `parent(X,Y)` and `person(X)` enumerates all the `descendant(X,Y)`

Summary of Answer Set Programming

- Provided a rationale for declarative problem solving
- Introduced logic programming under the answer set semantics
- Demonstrated that complex problems can easily be encoded in ASP

Mapping Institutions to ASP

- Smodels syntax (as seen earlier)
- Time instances to indicate state transitions
- Atoms:
  - `evtype(E,T)` describes the type of an event
  - `instant(I)` denote time instances
  - `final(I)` denotes the last time instance in a trace
  - `before(I1,I2)` and `next(I1,I2)` denote time order
  - `occurred(E,I)` indicates E happened at time I
  - `observed(E,I)` indicates E was observed at time I
  - `holdsat(P,I)` indicates that P holds at time I
  - `initiated(P,I)` and `terminated(P,I)` indicate that P is initiated/terminated at time I

Parts of the Mapping

- Each mapping for institution $\mathcal{I}$ consists of two parts
  - P_{base}: institution independent
    - responsible for the occurrence of observed events
    - deals with obligations
    - assures inertia
  - $P^{*}_{\mathcal{I}}$ specific for the institution
    - event generation
    - state transition

The institution program P_{base} (I)

```
occurred(E,I)            :- observed(E,I).
holdsat(P,I2)            :- holdsat(P,I1), not terminated(P,I1),
                            next(I1,I2), instant(I1;I2).
holdsat(P,I2)            :- initiated(P,I1),
                            next(I1,I2), instant(I1;I2).
occurred(viol(E),I)      :- occurred(E,I),
                            not holdsat(perm(E),I),
                            event(E), event(viol(E)), instant(I).
occurred(V,I)            :- holdsat(obl(E,D,V),I), occurred(D,I),
                            event(E;D;V), instant(I).
terminated(obl(E,D,V),I) :- occurred(E,I),
                            holdsat(obl(E,D,V),I),
                            event(E;D;V), instant(I).
terminated(obl(E,D,V),I) :- occurred(D,I),
                            holdsat(obl(E,D,V),I),
                            event(E;D;V), instant(I).
```


The institution program P_{base} (II)

To constrain the answer set to those containing observable traces we add the following rules to P_{base}:

```
{observed(E,I)} :- evtype(E,obs), event(E), instant(I), not final(I).
          ev(I) :- observed(E,I), event(E), instant(I).
                :- not ev(I), instant(I), not final(I).
                :- observed(E1,I), observed(E2,I), E1! = E2, instant(I),
                   event(E1), event(E2).
```

Thus an observable event occurs at each time instant, while the last constraint ensures that each answer set has only one observable event at any time instant.

A Shorthand

- $EX(\_, I)$ to denote the translation of expression $X \in \mathcal{X}$ into the body of an ASP rule referring to time I.
- $EX(x_1 \wedge x_2 \wedge \ldots x_n, I)$ is translated as $\mathtt{EX}(\mathtt{x}_1, \mathtt{I}), \mathtt{EX}(\mathtt{x}_2, \mathtt{I}), \ldots, \mathtt{EX}(\mathtt{x}_n, \mathtt{I})$.
- $EX(\neg p, I)$ becomes **not** $EX(p, I)$
- $EX(p, I)$ is translated as `holdsat(p,I)`.
- Thus $EX(perm(callup), \neg perm(murder), pow(murder), I)$ becomes `holdsat(perm(callup))`, **not** `holdsat(perm(murder))`, `holdsat(pow(murder))`

The Institution Dependent Part $P^{*}_{\mathcal{I}}$

$$
\begin{array}{lll}
p \in \mathcal{F} & \Leftrightarrow & \mathtt{ifluent(p).} \\
e \in \mathcal{E} & \Leftrightarrow & \mathtt{event(e).} \\
e \in \mathcal{E}_{obs} & \Leftrightarrow & \mathtt{evtype(e,obs).} \\
e \in \mathcal{E}_{instact} & \Leftrightarrow & \mathtt{evtype(e,act).} \\
e \in \mathcal{E}_{viol} & \Leftrightarrow & \mathtt{evtype(e,viol).} \\
\mathcal{C}^{\uparrow}(X, e) = P & \Leftrightarrow & \forall p \in P \cdot \mathtt{initiated(p,I)} :- \mathtt{occurred(e,I)}, EX(X, I). \\
\mathcal{C}^{\downarrow}(X, e) = P & \Leftrightarrow & \forall p \in P \cdot \mathtt{terminated(p,I)} :- \mathtt{occurred(e,I)}, EX(X, I). \\
\mathcal{G}(X, e) = E & \Leftrightarrow & g \in E, \mathtt{occurred(g,I)} :- \mathtt{occurred(e,I)}, \\
 & & \quad \mathtt{holdsat(pow(e),I)}, EX(X, I). \\
p \in S_0 & \Leftrightarrow & \mathtt{holdsat(p,i_0).}
\end{array}
$$

Soundness and Completeness

- We still need to initialise the ASP with time
- We refer to this program as P^{n}:

$$
\begin{array}{lll}
0 < k < n & : & \mathtt{instant(i_k).} \\
 & & \mathtt{next(i_k, i_{k+1}).} \\
 & & \mathtt{final(i_n).}
\end{array}
$$

- Together P_{base}, $P^{*}_{\mathcal{I}}$ and P^{n} generate $P^{n}_{\mathcal{I}}$

Theorem

Let $\mathcal{I} = \langle \mathcal{E}, \mathcal{F}, \mathcal{C}, \mathcal{G}, \Delta \rangle$ be an institution with $P^{n}_{\mathcal{I}}$ its corresponding answer set program. Then, a one-to-one mapping exists between the ordered event traces of length n and the answer sets of $P^{n}_{\mathcal{I}}$.

War in ASP (1)

```
ifluent(atwar).

event(shoot).
event(startwar).
event(declaretruce).
event(callup).
event(conscript).
event(murder).
event(provoke).
event(viol(shoot)).
event(viol(startwar)).
event(viol(declaretruce)).
event(viol(callup)).
event(viol(conscript)).
event(viol(provoke)).
```

```
ifluent(obl(startwar,shoot,murder)).

evtype(shoot,obs).
evtype(startwar,obs).
evtype(declaretruce,obs).
evtype(callup,obs).
evtype(conscript,inst).
evtype(murder,inst).
evtype(provoke,obs).
evtype(viol(shoot),viol).
evtype(viol(startwar),viol).
evtype(viol(declaretruce),viol).
evtype(viol(callup),viol).
evtype(viol(conscript),viol).
evtype(viol(murder),viol).
evtype(viol(provoke),viol).
```


War in ASP (2)

```
initiated(obl(startwar,shoot,murder),I) :- occurred(provoke,I),instant(I),
                                            not holdsat(atwar,I).
                     initiated(atwar,I) :- occurred(startwar,I),instant(I),
                                            not holdsat(atwar,I).
               initiated(perm(shoot),I) :- occurred(conscript,I),instant(I).
            initiated(pow(conscript),I) :- occurred(startwar,I),instant(I).

          terminated(atwar,I) :- occurred(declaretruce,I),instant(I),
                                  holdsat(atwar,I).
    terminated(perm(shoot),I) :- occurred(declaretruce,I),instant(I).
terminated(pow(conscript),I) :- occurred(declaretruce,I),instant(I).
```


War in ASP (3)

```
occurred(conscript,I) :- occurred(callup,I),instant(I),
                          holdsat(pow(conscript),I).
   occurred(murder,I) :- occurred(viol(shoot),I),instant(I).
```

```
instant(i0;i1;i2;i3).
init(i0).
next(i0,i1).
next(i1,i2).
next(i2,i3).
final(i3).
```

```
holdsat(perm(callup),i0).
holdsat(perm(startwar),i0).
holdsat(perm(conscript),i0).
holdsat(perm(declaretruce),i0).
holdsat(perm(murder),i0).
holdsat(perm(provoke),i0).
holdsat(pow(murder)),i0.
```


Queries

- Given an institutional specification in ASP, queries are possible:
  - Given some known initial state, and a complete trace of events, what is the current social state?
  - Given partial information about the initial and/or current state what are the possible sequences of events which led us to this state.
- Two rules need to be added:
  - one to represent the query
  - one to indicate to the solver that we are only interested in those ordered traces that satisfies the condition

Example

```
condition :- holdsat(obl(startwar,shoot,murder),I),instant(I).
compute all {condition}.
```


Querying Exercise

Example

Reconsider the borrowing institution from the previous section. Write a query that returns all traces in which agent *a* leaves the institution without paying back.

Summary of Single Institutions

- To verify and reasoning about institutions, we mapped the formal model to an answer set program
- The answer sets of program correspond to the traces of the institution

Reading material

Chitta Baral.
Knowledge Representation, Reasoning and Declarative Problem Solving.
Cambridge Press, 2003.

M. Gelfond and V. Lifschitz.
The stable model semantics for logic programming.
In *Proc. of fifth logic programming symposium*, pages 1070–1080. MIT PRESS, 1988.

Vladimir Lifschitz.
Answer set programming and plan generation.
Journal of Artificial Intelligence, 138(1-2):39–54, 2002.

Part V

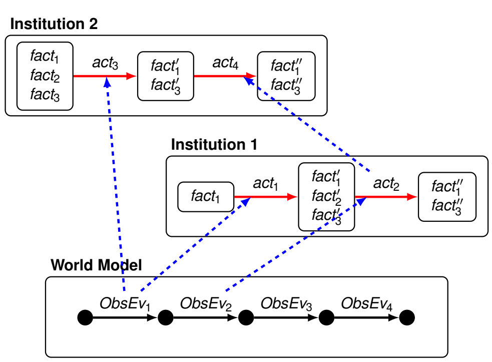
Design and Enactment

Outline

Allowing for Multi-Institutions

- Several institutions that
  - operate in the same environment
  - possible can influence each other directly or indirectly
- extend the formal framework to allow this

Specification Model



We would like

- to allow institutions to
  - generate events in another institution
  - to initialise and terminate fluents in another institution
- but only if this institution has permission to do so.
- so we need
  - alter our permission facts to be able to specify institutions
  - introduce permissions to initialise and terminate fluents
- this has an effect on
  - the event generation $\mathrm{GR}(S, E)$
  - initiation $\mathrm{INIT}(S, e_{obs})$
  - termination $\mathrm{TERM}(S, e_{obs})$

Extended and New Institutional Fluents

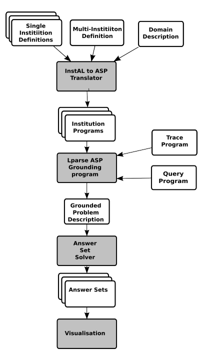
Definition

- Multi-institution: $\mathcal{M} = \langle \mathcal{I}_1, \ldots, \mathcal{I}_n \rangle$
  - $\mathcal{W}_i$: pow$(j, e) : 1 \leq j \leq n, e \in \mathcal{E}^j_{instact}$
    - Institution j has the power to bring about action e in institution i.
  - $\mathcal{S}_i$: inipow(j, f): $1 \leq j \leq n$, where $f \in \mathcal{D}_i$
    - Institution j is empowered to initiate some domain fluent f in institution i
  - $\mathcal{T}_i$: termpow(j, f): $1 \leq j \leq n$, where $f \in \mathcal{D}_i$
    - Institution j is empowered to terminate some domain fluent f in institution i
- $\mathcal{F}_{\mathcal{M}} = \bigcup_{j=1}^{n} \mathcal{F}_j$
- $\Sigma_{\mathcal{M}} = \Sigma_1 \times \ldots \times \Sigma_n$.

Support for the Designer

- Primary objective is to specify the behaviour of an institution in terms of its norms, and ...
- To be able to test the properties of the model
- ASP code can be useful, but ...
- contains low level details that can hinder the design process
- Furthermore, lots of ASP can be created almost automatically
- $\Rightarrow$ a domain-specific event language may be an appropriate design medium
- We define a (multi-)institution specific action language
- InstAL

InstAL Workflow



- Write specification in InstAL
- Generate ASP
- Combine with trace and query
- Compute grounding
- Generate answer sets
- Vizualise results

The Dutch Auction

- One agents acts as auctioneer
- One or more agents play the bidders
- The purpose of the protocol as a whole is either to determine a winning bidder and a valuation for a particular item on sale, or to establish that no bidders wish to purchase the item.

The Protocol (I)

Definition

- Round starts: Auctioneer selects a price for the item and informs each of the bidders present of the starting price. The auctioneer then waits for a given period of time for bidders to respond.
- Upon receipt of the starting price, each bidder has the choice as to whether to send a message indicating their desire to bid on the item at that price, or to send no message indicating that they do not wish to bid on the item.
- At the end of the prescribed period of time, if the auctioneer has received a single bid from a given agent, then the auctioneer is obliged to inform each of the participating agents that this agent has won the auction.

The Protocol (II)

Definition

- If no bids are received at the end of the prescribed period of time, the auctioneer must inform each of the participants that the item has not been sold.
- If more than one bid was received then the auctioneer must inform each agent that a conflict has occurred.
- In the case where the item is sold or unsold, the protocol is finished.
- In the case where a conflict occurs then the auctioneer must re-open the bidding and start the round again in order to resolve the conflict.

DAR InstAL(I)

Example

```
institution dutch;
type Bidder;
type Auct;

create event createdar;

exogenous event priceto;
exogenous event bidto;
exogenous event desto;

exogenous event annprice(Auct,Bidder);
exogenous event annbid(Bidder,Auct);
exogenous event annconf(Auct,Bidder);
exogenous event annsold(Auct,Bidder);
exogenous event annunsold(Auct,Bidder);

inst event pricedl;
inst event biddl;
inst event desdl;
inst event desdl;
```


DAR InstAL(II)

Example

```
inst event price(Auct,Bidder);
inst event bid(Bidder,Auct);
inst event conf(Auct,Bidder);
inst event sold(Auct,Bidder);
inst event unsold(Auct,Bidder);

dest event badgov;
dest event finished;

inst event alerted(Bidder);

fluent onlybidder(Bidder);
fluent havebid;
fluent conflict;

initially pow(price(A,B)), perm(price(A,B)),
    perm(annprice(A,B)),perm(badgov),pow(badgov),
    perm(pricedl),pow(pricedl), perm(priceto),
    perm(biddl),perm(bidto),perm(desto);
```


DAR InstAL(III)

Example

```
-(Phase 1:  pricing) -
initially obl(price(A,B),pricedl,badgov);
annprice(A,B) generates price(A,B);
price(A,B) terminates pow(price(A,B));
price(A,B) initiates pow(bid(B,A)),perm(bid(B,A)),perm(annbid(B,A));

- (Phase 2:  bidding) -
annbid(A,B) generates bid(A,B);
bid(B,A) terminates pow(bid(B,A)),perm(bid(B,A)),perm(annbid(B,A));
bid(B,A) initiates havebid,onlybidder(B) if not havebid;
bid(B,A) terminates onlybidder(_) if havebid;
bid(B,A) initiates conflict if havebid;

s - (Phase 3:  Resolution) -
annsold(A,B) generates sold(A,B);
annunsold(A,B) generates unsold(A,B);
annconf(A,B) generates conf(A,B);
biddl terminates pow(bid(B,A));
```


DAR InstAL(IV)

Example

```
- (Phase 3:  Resolution cont.)  -
biddl initiates pow(sold(A,B)),pow(unsold(A,B)),
    pow(conf(A,B)), pow(alerted(B)),perm(alerted(B));

biddl initiates perm(annunsold(A,B)),perm(unsold(A,B)),
    obl(unsold(A,B),desdl,badgov) if not havebid;
biddl initiates perm(annsold(A,B)),perm(sold(A,B)),
     obl(sold(A,B), desdl, badgov) if havebid, not conflict;
biddl initiates perm(annconf(A,B)),perm(conf(A,B)),
     obl(conf(A,B), desdl, badgov) if havebid, conflict;
unsold(A,B) generates alerted(B);
sold(A,B) generates alerted(B);
conf(A,B) generates alerted(B);
alerted(B) terminates pow(unsold(A,B)), perm(unsold(A,B)),
    pow(sold(A,B)), pow(conf(A,B)), pow(alerted(B)),
    perm(sold(A,B)), perm(conf(A,B)), perm(alerted(B)),
    perm(annconf(A,B)),perm(annsold(A,B)),perm(annunsold(A,B));

desdl generates finished if not conflict;
```


DAR InstAL(V)

Example

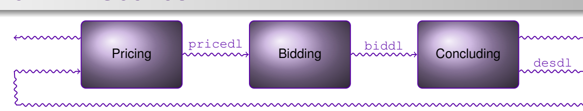
```
- (Phase 3:  Resolution cont.)  -
desdl terminates havebid,conflict,perm(annconf(A,B));
desdl initiates pow(price(A,B)), perm(price(A,B)),
     perm(annprice(A,B)), perm(pricedl),pow(pricedl),
     obl(price(A,B),pricedl,badgov) if conflict;

priceto generates pricedl;
pricedl terminates pow(pricedl);
pricedl initiates pow(biddl);

bidto generates biddl;
biddl terminates pow(biddl);
biddl initiates pow(desdl);

desto generates desdl;
desdl terminates pow(desdl);
```


The DAR Scenes

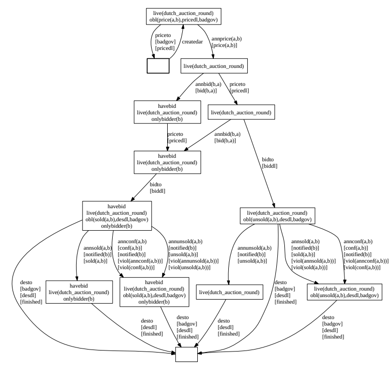


- The DAR protocol is governed by the deadlines InstALpricedl, biddl and InstALdesdl.
- When the deadline event occurs, auctioneer and bidder are given different powers and permissions that allow the protocol to proceed to a different scene
- For example, InstALpricedl announces the end pricing scenes and provides the bidders with the power and permission to start bidding
- InstALdesdl is special in the sense that it either marks the end of the protocol (bidding was successfull) or has to start all over again.

Verification

- Just add the time frame
- and if requested a query program
- Run the solver and obtain the answer sets
- By varying the time frame you obtain all the states of the the protocol

The DAR Protocol as Finite State Machine



References

Owen Cliffe, Marina De Vos, and Julian Padget.
Embedding landmarks and scenes in a computational model of institutions.
In *Proceedings of the Multi-Agent Logics, Languages and Organizations Federated Workshops, COIN workshop*, Durham, UK, September 2007.

Owen Cliffe, Marina De Vos, and Julian Padget.
Specifying and reasoning about multiple institutions.
In Pablo Noriega, Javier Vzquez-Salceda, Guido Boella, Olivier Boissier, Virginia Dignum, Nicoletta Fornara, and Eric Matson, editors, *Coordination, Organization, Institutions and Norms in Agent Systems II - AAMAS 2006 and ECAI 2006 International Workshops, COIN 2006 Hakodate, Japan, May 9, 2006 Riva del Garda, Italy, August 28, 2006*, volume 4386 of *Lecture Notes in Computer Science*, pages 67–85. Springer Berlin / Heidelberg, 2007.

Michael Gelfond and Vladimir Lifschitz.
Action languages.
Electron. Trans. Artif. Intell., 2:193–210, 1998.

Part VI

Summary

Outline

Related Work (I)

- Colombetti et al in [2]
  - abstract model based on social commitments
  - registration and interaction rules, authorisations and an ontology
  - builds on CTL± extension of CTL[1]
  - uses past tense modalities for reasoning about actions that have occurred
- Dignum in [3] uses CTL for representing contracts for agent organisations
- Event Calculus (EC) [5]
  - [6] uses EC for the behaviour of commitments
  - [4] uses EC for general social models
  - In ASP, induction, abduction, non-monotonicity and defaults come for free (no circumscription).
  - no even generation in EC
  - inertia is not axiomatic in ASP

References

E. M. Clarke, E. A. Emerson, and A. P. Sistla.
Automatic verification of finite-state concurrent systems using temporal logic specifications.
ACM Transactions on Programming Languages and Systems, 8(2):244–263, 1981.

M. Colombetti, N. Fornara, and M. Verdicchio.
The role of institutions in multiagent systems.
In *Proceedings of the Workshop on Knowledge based and reasoning agents, VIII Convegno AI*IA 2002, Siena, Italy*, 2002.

Virginia Dignum, John-Jules Meyer, Frank Dignum, and Hans Weigand.
Formal Specification of Interaction in Agent Societies.
In *Formal Approaches to Agent-Based Systems (FAABS-02)*, volume 2699 of *Lecture Notes in Computer Science*, pages 37–52, October 2003.

L. Kamara, A. Artikis, B. Neville, and J. Pitt.
Simulating computational societies.
In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Proceedings of workshop on engineering societies in the agents world (esaw)*, LNCS 2577, pages 53–67. Springer, 2003.

R Kowalski and M Sergot.
A logic-based calculus of events.
New Gen. Comput., 4(1):67–95, 1986.

Pinar Yolum and Munindar P. Singh.
Flexible protocol specification and execution: applying event calculus planning using commitments.
In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 527–534. ACM Press, 2002.

Related Work (II)

- Artikis et al. in [1, 2, 5]
  - specification of a normative social system
  - power, empowerment and obligation
  - use EC and a subset of the action language $C+$[4]
  - no violations as events
  - our obligations are deadline-sensitive and can raise obligations
- Sergot in [6]
  - action language for normative systems
  - same reservations as above
- Buccafurri et al. in [3]
  - uses social logic programs to denote acceptable and unacceptable states
  - agent perspective vs institution perspective

References

A. Artikis, M. Sergot, and J. Pitt.
An executable specification of an argumentation protocol.
In *Proceedings of conference on artificial intelligence and law (icail)*, pages 1–11. ACM Press, 2003.

A. Artikis, M. Sergot, and J. Pitt.
Specifying electronic societies with the Causal Calculator.
In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Proceedings of Workshop on Agent-Oriented Software Engineering III (AOSE)*, LNCS 2585. Springer, 2003.

Francesco Buccafurri and Gianluca Caminiti.
A social semantics for multi-agent systems.
In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *LPNMR*, volume 3662 of *Lecture Notes in Computer Science*, pages 317–329. Springer, 2005.

Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner.
Nonmonotonic causal theories.
Artificial Intelligence, Vol. 153, pp. 49-104, 2004.

L. Kamara, A. Artikis, B. Neville, and J. Pitt.
Simulating computational societies.
In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Proceedings of workshop on engineering societies in the agents world (esaw)*, LNCS 2577, pages 53–67. Springer, 2003.

Marek Sergot.
(C+)++: AN ACTION LANGUAGE FOR REPRESENTING NORMS AND INSTITUTIONS.
Technical report, Imperial College, London, August 2004.

Summary

- Agent Societies: case for the norm-regulated agent and for an agent being governed by multiple interacting institutions
- Answer Set Programming: a logic programming paradigm that supports the definition of domain-oriented models, checking and querying
- Single Institutions: a trace-based formalization of a single institution in an executable framework
- Multi-Institutions: a direct extension of the single institutional model to account for inter-institutional events
- Modelling Behaviour: a demonstration of the application of the formalization to some familiar scenarios

Open Research Issues

- Scalability: effect of realistic numbers of interacting institutions on computational cost
- Modularity: can the multi-institution work in practice and how are institutional conflicts to be resolved
- On-line reasoning for norm-aware and non-norm-aware agents
- Interaction with real legal frameworks
- Accessibility: for agents that cannot process the normative information

Towards Organisational-oriented Programming Electronic Institutions Development Environment

Marc Esteva

Artificial Intelligence Research Institute (IIIA-CSIC), Spain

eMarkets Group, University of Technology Sydney, Australia

Overview

EI: Motivation

- **Open multi-agent systems** are populated by *heterogeneous*, *self-interested* agents, developed by different people, using different languages and architectures. Participants *change* over time and are unknown in advance.
- With the expansion of the Internet open multi agent systems represent the most important area of application of multi agent systems.

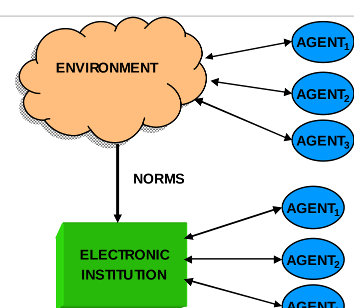
EI: Motivation

- Institutions have proved to successfully regulate human societies for a long time:
  - created to achieve particular goals while complying norms.
  - responsible for defining the rules of the game (norms), to enforce them and assess penalties in case of violation.
- Examples: auction houses, parliaments, stock exchange markets,….

EI: Motivation

- *Research issue*: **methodologies** and **software tools** to support their **design, verification, development, and analysis**.
- *Goal*: **principled design and development of open multi agent systems.**

Approach



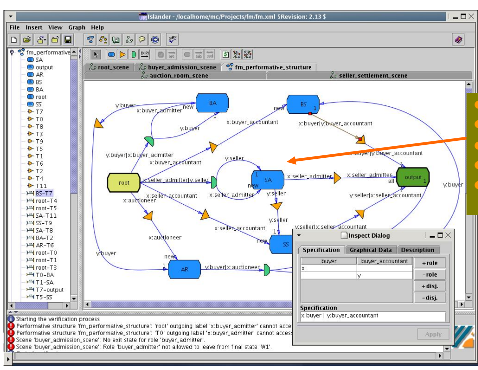
Institutions in the sense proposed by North "… set of artificial constraints that articulate agent interactions".

Approach

- **Electronic institutions development** can be divided into two basic steps:
  - **Formal specification** of institutional rules.
  - **Execution** via an infrastructure that **mediates** agents' interactions while **enforcing** the institutional rules.
- The **formal specification** focuses on **macro-level** (rules) aspects of agents, not in their micro-level (players) aspects.
- The **infrastructure** is required to be of **general purpose** (can *interpret* any formal specification).

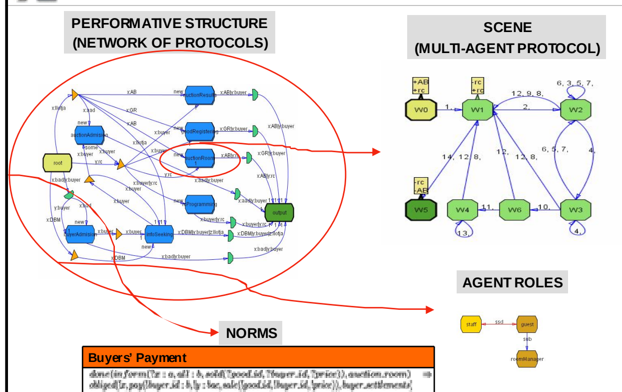
Overview

Electronic Institution Specification with ISLANDER



- **Network of protocols**
- **Multi-agent protocols**
- **Norms**
- **Agent Roles**
- **Common Ontology and language**

Electronic Institution Components



The ("Hello World") Chat Example

- A simple institution where agents interact simulating a chat.
- Each agent owns a main topic and a list of subtopics he is interested in.
- Agents create a chat room per main topic.
- They can join the scenes created by other agents
- The institution keeps track of active chat rooms to provide information to agents.

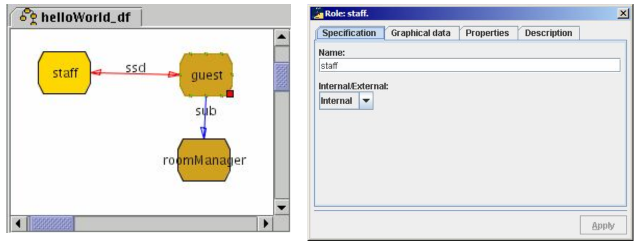
Dialogical Framework Components

- Common ontology
- Valid communication language expressions
  - List of illocutionary particles
  - Content language
- Roles that agents can play
  - Internal Roles
  - External Roles
- Role relationships

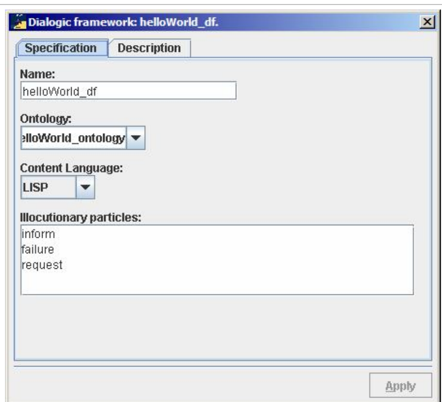
Roles

- Each role defines a pattern of behaviour within the institution (actions associated to roles).
- Agents can play multiple roles at the same time
- Agents can change their roles.
- Two types of roles:
  - Internal: played by the staff agents to which the institution delegates its services and tasks.
  - External: played by external agents.
- Role relationships:
  - Static incompatibility (ssd)
  - Dynamic incompatibility (dsd)
  - Hierarchy (sub)
- Information model per role: a set of attributes that define the information that the institution keeps per each role.

Chat roles



Chat Dialogic Framework



Communication Language

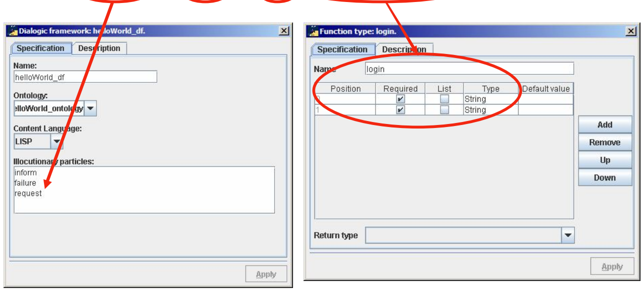
- CL expressions are formulae of the form $(i\ (\alpha_i\ r_i)\ \beta\ \gamma\ \tau)$ where:
  - i is an illocutionary particle (e.g. request, inform);
  - α_i can be either an agent variable or an agent identifier;
  - r_i can be either a role variable or a role identifier;
  - β represents the addressee(s) of the message and can be:
    - $(\alpha_k\ r_k)$ the message is addressed to a single agent.
    - r_k the message is addressed to all the agents playing role r_k.
    - "all" the message is addressed to all the agents in the scene.
  - γ is an expression in the content language.
  - τ can be either a time variable or a time-stamp

- *(request (?x guest) (!y staff) (login ?user ?email))*

Communication Language

Roles

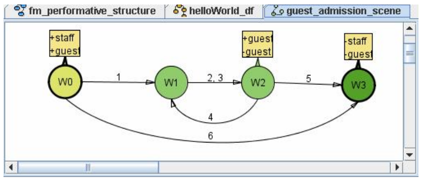
(request (?x guest) (!y staff) (login ?user ?email))

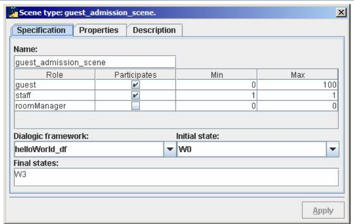


Scenes

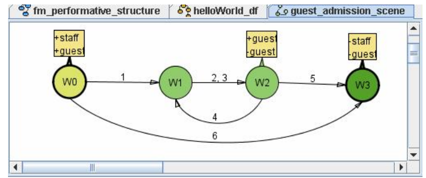
- Specification level
  - A scene is a pattern of multi-agent interaction.
  - Scene protocol specified by a finite state oriented graph where the nodes represent the different states and oriented arcs are labelled with *illocution schemes* or *timeouts*.
- Execution level
  - Agents may join or leave scenes.
  - Each scene keeps the *context* of its multi-agent interaction.
  - A scene can be multiply executed and played by different groups of agents.

Guest admission scene





Guest admission scene. Illocutions



1. (request (?x guest) (?y staff) login(?user ?email)))
2. (inform (!y staff) (!x guest) accept()))
3. (failure (!y staff) (!x guest) deny(?code)))
4. (request (?x guest) (!y staff) login(?user ?email)))
5. (inform (!y staff) (all guest) close()))
6. (inform (?y staff) (all guest) close()))

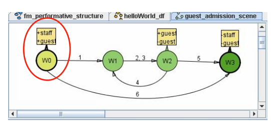
Scenes

- Illocution schemes: at least the terms referring to agents and time are variables.
- Semantics of variable occurrences:
  - ?x: variable x can be bound to a new value.
  - !x: variable x must be substituted by its last bound value.

  Example:

  (request (?x guest) (!y staff) login(?user ?email)))
- **Context** of a conversation captured on a list of variables' bindings.

Guest admission scene. Illocutions



1. (request (?x guest) (?y staff) login(?user ?email))

6. (inform (?y staff) (all guest) close()))

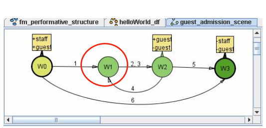
- Two agents in the scene: John as guest and Mike as staff
- Agent John utters an illocution:

  (request (John guest) (Mike Staff) login(John john@hotmail.com))

- The illocution matches arc 1 and the scene evolves to W1.
- Substitutions:

  [?x/John, ?y/Mike, ?user/John, ?email/john@hotmail.com]

Guest admission scene. Illocutions



2. (inform (!y staff) (!x guest) accept()))
3. (failure (!y staff) (!x guest) deny(?code)))

- Former bindings:

  [?x/John, ?y/Mike, ?user/John, ?email/john@hotmail.com]

- Only illocutions matching the following schemes will be valid:

  (inform (Mike staff) (John guest) accept()))

  (failure (Mike staff) (John guest) deny(?code)))

Scene Constraints

- Constraints capture how past actions in a scene affect its future evolution:
  - restricting the valid values for a variable
  - restricting the paths that a conversation can follow
- Examples:
  - A buyer can only submit a single bid at auction time.
  - A buyer must submit a bid greater than the last one.
  - An auctioneer can not declare a winner if two buyers have submitted a bid at the higher value.
  - An agent can not repeat an offer during a negotiation process.

Scene

- Example
  - Illocution scheme:
    commit((?y buyer) (!x auctioneer) bid(!good_id, ?price))

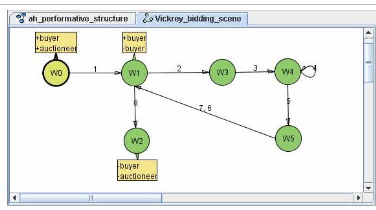
    ?price $\in$ (0.+∞)

  - Constraint:
    (> ?price !starting_price)
    ?price $\in$ (!starting_price.+∞)

Variable Occurrences

- ?x: binding occurrence
- !x: stands for the last binding of variable x.
- !x ($w_i w_j$): stands for the bindings of variable x the last time that the conversation evolve from $w_i w_j$.
- !x ($w_i w_j$ i): stands for the bindings of variable x the last i times that the conversation evolved from $w_i w_j$.
- !x ($w_i w_j$ *): stands for the bindings of variable x all the times that the conversation evolved from $w_i w_j$.

Example: Vickrey auction



1 (inform (?x auctioneer) (all buyer) startauction(?a))

2 (inform (!x auctioneer) (all buyer) startround(?good ?price ?bidding_time))

3 (inform (!x auctioneer) (all buyer) offer(!good !price))

4 (request (?y buyer) (!x auctioneer) bid(!good ?bid_price))

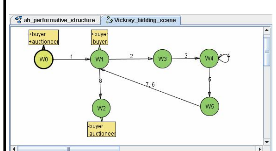
5 [!bidding_time])

6 (inform (!x auctioneer) (all buyer) sold(!good ?sold_price ?buyer_id))

8 (inform (!x auctioneer) (all buyer) close())

7 (inform (!x auctioneer) (all buyer) withdrawn(!good))

Constraints



1 (inform (?x auctioneer) (all buyer) startauction(?a))

2 (inform (!x auctioneer) (all buyer) startround(?good ?price ?bidding_time))

3 (inform (!x auctioneer) (all buyer) offer(!good !price))

4 (request (?y buyer) (!x auctioneer) bid(!good ?bid_price))

5 [!bidding_time]

6 (inform (!x auctioneer) (all buyer) sold(!good ?sold_price ?buyer_id))

8 (inform (!x auctioneer) (all buyer) close())

7 (inform (!x auctioneer) (all buyer) withdrawn(!good))

Constraint :

?y $\notin$!y (w3 w4)
?bid_price > !price

Performative Structure

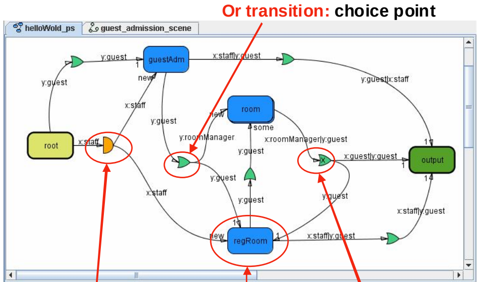
- Complex activities can be specified by establishing relationships among scenes that define:
  - causal dependency (e.g. a guest agent must go through the admission scene before going to the chat rooms)
  - synchronisation points (e.g. synchronise a buyer and a seller before starting a negotiation scene)
  - parallelisation mechanisms (e.g. a guest agent can go to multiple chat rooms)
  - choice points (e.g. a buyer leaving an admission scene can choose which auction scene to join)
  - the role flow policy

Performative Structure

- Performative structures as networks of scenes.
- Transitions to link scenes.
- Arcs connecting scenes and transitions labelled with constraints and roles.
- Agents moving from a transition to a scene may join *one*, *some* or *all* current executions of the target scene(s) or start new executions.
- The specification allows to express that simultaneous executions of a scene may occur.

Chat Performative Structure

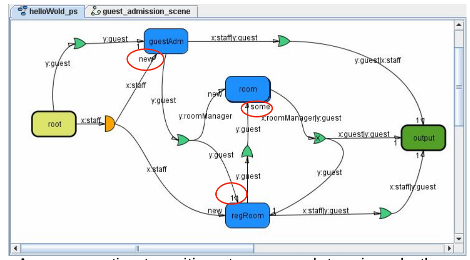
Or transition: choice point



Scene XOr transition: exclusive choice point

And transition: synchronisation and parallelisation point

Chat Performative Structure

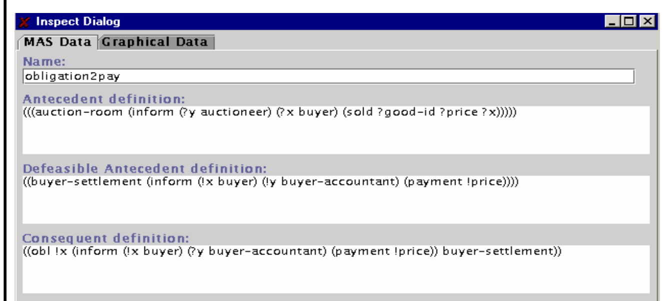


Arcs connecting transitions to scenes determine whether agents join *one*, *some* or *all* current executions of the target scene(s) or whether *new* executions are started.

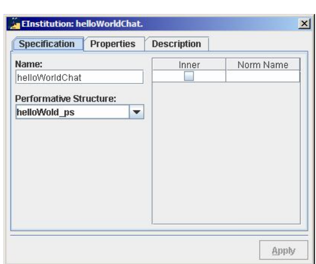
Norms

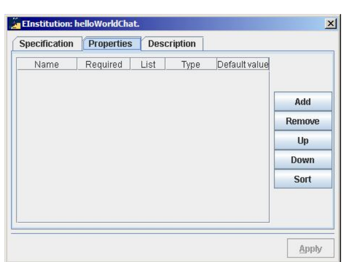
- Norms define the consequences of agents actions within the institution.
- Such consequences are captured as obligations.
  - Obl(x,Φ,s): meaning that agent x is obliged to do Φ in scene s.
- Norms are a special types of rules specified by three elements:
  - Antecedent: the actions that provoke the activation of the norm and boolean expressions over illocution scheme variables.
  - Defeasible antecedent: the actions that agents must carry out in order to fulfil the obligations.
  - Consequent: the set of obligations
- Actions expressed as pairs of scene and illocution schema.

Norms



Electronic Institutions Definition

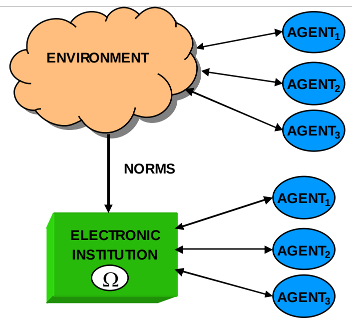




Overview

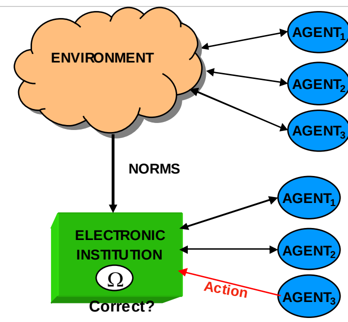
I. Introduction
II. Specifying electronic institutions
III. Running electronic institutions
IV. Electronic Institutions Development Environment
V. 3D Electronic Institutions
VI. Conclusions

Electronic institution execution



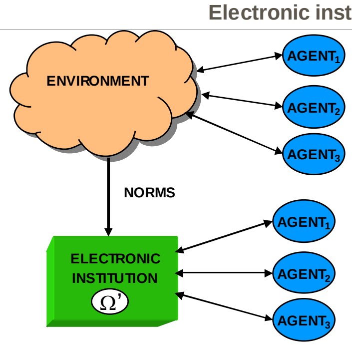
EXECUTION STATE: Ω

Electronic institution execution



EXECUTION STATE: Ω

Electronic institution execution

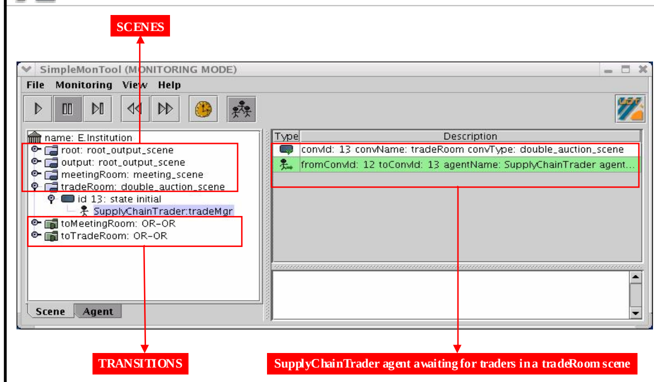


EXECUTION STATE: Ω -> Ω'

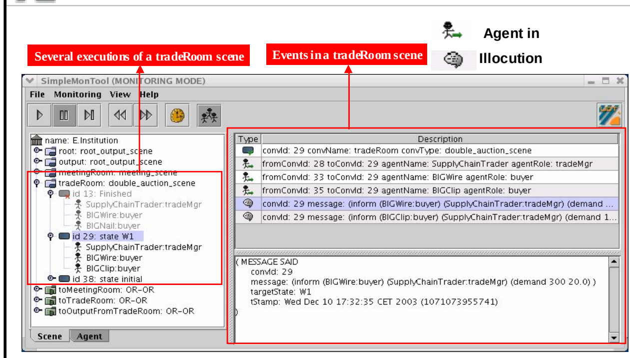
Electronic institution execution

- Electronic institutions are populated at run-time by heterogenous, self-interested agents.
- The institution execution can be regarded as the execution of its different scenes (processes).
- Agents move from scene to scene.
- Agents interact within scenes via speech acts.
- Agents acquire and fulfil obligations.

Electronic institution execution monitoring



Electronic institution execution monitoring



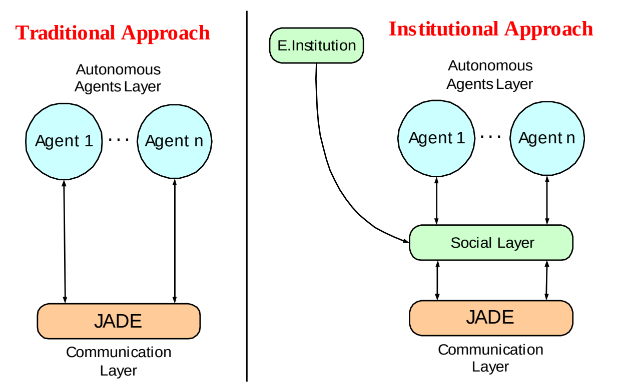
Execution State

- $\Omega = \langle Ag, \Sigma, T, Obl \rangle$ stands for an **institution execution state** where:
  - $Ag = \{ag_1, ..., ag_n\}$ is a finite set of participating agents.
  - $\Sigma = \{ \sigma_i^k \mid s_i \in S, k \in N\}$ is the set of all scene executions.
  - $T = \{T_1, ..., T_n\}$ stands for all transitions executions.
  - $Obl \{obl_1, ..., obl_n\}$ is the set of agents' pending obligations.
- $\sigma_i^k = \{\omega, B, A\}$ stands for **scene execution state** where:
  - ω represents the scene's current state.
  - $B = \{\beta_1, ..., \beta_n\}$ stands for the context (bindings) produced by illocutions.
  - $A = \{(ag,r) \mid ag \in Ag, r \in R\}$ is the set participating agents along with their roles.
- Each **transition execution state** $T_i = \{ (ag, \delta) \mid ag \in Ag, \delta = \{ (\sigma_i^k, r) \mid \sigma_i^k \in \Sigma, r \in R\}\}$ contains agents' target scenes.

Infrastructure operations

Specification	Functionality
Institution	$enter(ag, Roles)$ $exit(ag)$
Performative structure	$create\_scene(s)$ $close\_scene(\sigma)$
Scene	$join(\sigma, SAgents)$ $update\_state(\sigma, \iota)$ $update\_state(\sigma, \tau)$ $leave(\sigma, SAgents)$
Transition	$add\_agents(t, TAgents)$ $move\_to(t, ag, Target)$ $fire(t)$ $remove\_agents(t, TAgents)$
Norm	$add\_obligations(Obligations)$ $remove\_obligations(Obligations)$

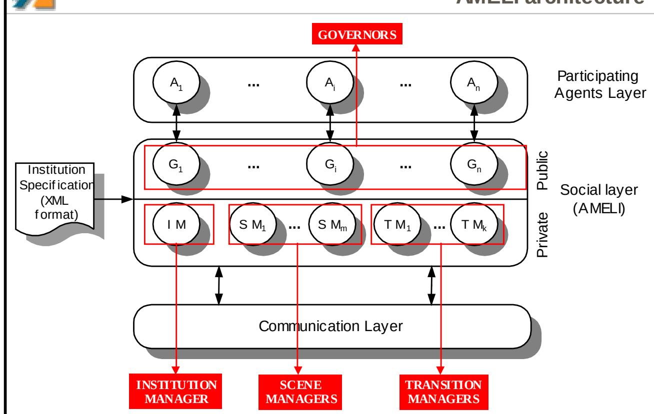
Electronic Institution Infrastructure



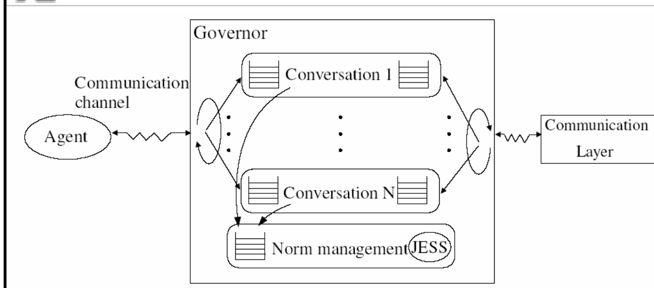
AMELI functionalities

- **MEDIATION**
  - To facilitate agent communication within scenes.
- **COORDINATION AND ENFORCEMENT**
  - To guarantee the correct evolution of each scene.
  - To guarantee legal movements between scenes.
  - To control the obligations participating agents acquire and fulfil.
- **INFORMATION MANAGEMENT**
  - To facilitate the information agents need in the institution.

AMELI architecture



Governor



- Mediates between institution and participating agent.
- Controls that an agent behaves according to the institution specification (rules).

Agent to Governor Messages

Action	Description
enterInstitution	Request to enter the institution
moveToTransition	Request to move from a scene to a transition
moveToScenes	Request to move from a transition to several scenes
saySceneMessage	Request to say a message in a scene
accesScenes	Ask for the scenes the agent can join from a transition
accesTransitions	Ask for the transitions the agent can join from a scene
agentObligations	Ask for pending obligations
sceneState	Ask for a scene's current state
scenePlayers	Ask for agents in a scene

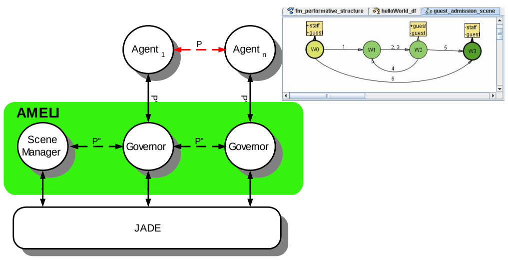
Governor to Agent Messages

Action	Description
enteredInstitution	The agent has entered the institution
exitedInstitution	The agent leaves the institution
enteredInstitutionFailed	The agent could not enter the institution
saidSceneMessage	An agent message has been said within a scene
saySceneMessageFailed	Agent message in a scene has failed
receivedSceneMessage	Reception of a message for the agent within a scene
timeoutTransition	The scene state has evolved as a consequence of the expiration of a timeout
enteredAgent	An agent has entered the scene
exitedAgent	An agent has left the scene
finishedScene	The scene has finished
currentAccesScenes	List of all the scenes that the agent can move into from a transition
movedToScene	The agent has entered a scene
moveToSceneFailed	Agent attempt to move to a scene failed
currentAccessTransitions	Informs of all the transitions that the agent can move into from a scene
movedToTransition	The agent has been moved to a transition
moveToTransitionFailed	Agent attempt to move to a transition failed
acquiredObligations	Informs of acquired obligations by the agent
obligationsFulfilled	Informs of fulfilled obligations by the agent
currentObligations	Informs about the current obligations of the agent
currentSceneState	Informs about the scene state
currentScenePlayers	Informs about the agents within a scene

Scene execution

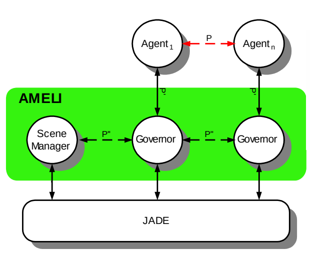
- Agents make a scene execution evolve by uttering illocutions which are correct with respect to the scene specification and its run-time "context".
- A scene manager updates a scene execution state after:
  - Validating an agent's illocution
  - Authorising agents to join or leave
  - Time-out expirations
- The scene manager and participating agents' governors coordinate to guarantee the correct evolution of the scene execution.

Scene Execution



Protocol P: scene protocol played by the participating agents and mediated by governors.

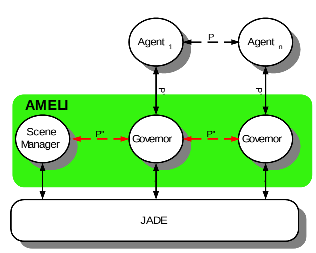
Scene Execution



Action	Description
enterInstitution	Request to enter the institution
moveToTransition	Request to move from a scene to a transition
moveToScenes	Request to move from a transition to several scenes
saySceneMessage	Request to say a message in a scene
accesScenes	Ask for the scenes the agent can join from a transition
accesTransitions	Ask for the transitions the agent can join from a scene
agentObligations	Ask for pending obligations
sceneState	Ask for a scene's current state
scenePlayers	Ask for agents in a scene

Protocol P': protocol between participating agents and governors.

Scene Execution



Protocol P'': coordination protocol between scene managers and governors.

Transition management

- Each transition is managed by a transition manager.
- Agents within a transition can ask for target scenes to join.
- The transition manager is in charge of controlling when the transition can be fired (agents can move).

Norm management

- Norms managed as a rule-based system.
- Constructed from an ISLANDER specification.
- The facts are illocutions uttered by agents.
- Each governor manages his agent’s obligations.

Norm management. 1st Generation

Antecedent

A norm N_i :

$$(s_1, \gamma_1) \wedge \ldots \wedge (s_m, \gamma_m) \wedge e_1 \wedge \ldots \wedge e_k \wedge$$

Defeasible Antecedent

$$\wedge \neg (s_{m+1}, \gamma_{m+1}) \wedge \ldots \wedge \neg (s_{m+n}, \gamma_{m+n}) \rightarrow obl_1 \wedge \ldots \wedge obl_p$$

Obligations

is transformed into:

Norm Activation

$$R1_i : (s_1, \gamma_1) \wedge \ldots \wedge (s_m, \gamma_m) \wedge e_1 \wedge \ldots \wedge e_k \rightarrow$$
$$assert(obl_1 \ldots obl_p) \wedge addRule(R2'_i, RB)$$

Obligations fulfilment

$$R2_i : (s_{m+1}, \gamma_{m+1}) \wedge \ldots \wedge (s_{m+n}, \gamma_{m+n}) \rightarrow$$
$$retract(obl_1 \ldots obl_p) \wedge dropRule(R2_i, RB)$$

Norm management. 2nd Generation

- Based on embedding the formal model of norms by Dignum et al. (EUMAS 05)
- Their approach adds conditional, temporal and precedence notions to obligations:

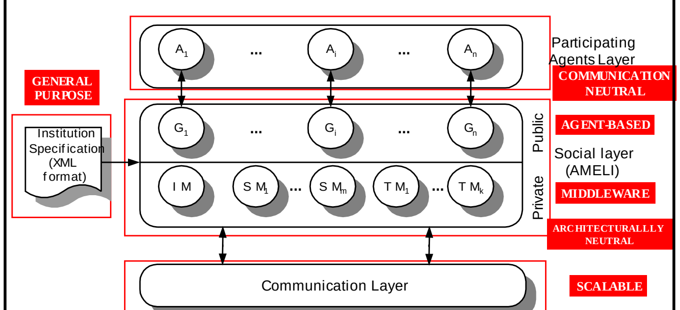
$$\text{OBLIGED}((register\_admin \text{ DO } correct(data)) \text{ IF } (incorrect(data)))$$

$$\text{OBLIGED}((allocator \text{ DO } assign(heart, recipient)) \text{ BEFORE } (time(done(extraction(heart, donor))) + 6hours))$$

Norms – The Jess implementation

- The main idea of our approach is the conversion of Dignum’s norms into Jess rules.
- We classify norms into:
  - **conditional** *(IF clause)*,
  - **time-dependent** *(BEFORE, AFTER or BETWEEN clause followed by a date or time period)*
  - **action-dependent** *(BEFORE, AFTER or BETWEEN clause followed by an action).*

AMELI implementation features



Overview

Are Electronic Institutions Enough?

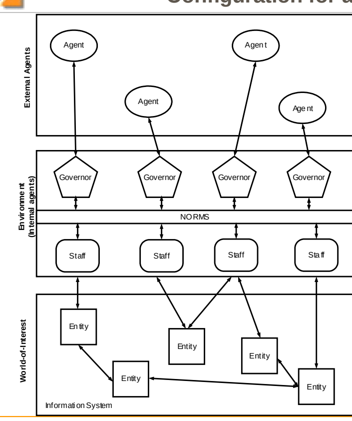
Let's summarise first !

- EIs enact norms to structure the environment.
- Norms can be thought as physical laws or as social conventions that shape/constrain the evolution of interactions.
- Dynamics of the environment restricted to those that satisfy the social laws represented by norms and enacted by the coordinated actions of governors and staff agents.
- The flexibility of EIs comes from its clear separation of concerns between the internal behaviour of agents and their external interactions (**environment modeling**).

Are Electronic Institutions Enough?

- The environment is given structure so that agents have an easy comprehension of its working laws.
- These restrictions help a lot in the programming of agents by restricting the set of actions agents have to consider at each moment in time.
- And yet...
- ...MAS applications are usually concerned with some external world-of-interest (WoI) in addition to the agent society issues.
- The WoI is application-specific and refers to the part of the world that is relevant to the application. F.i. for a climate control application, the WoI comprises rooms, doors, heaters, etc.

Configuration for a MAS application using EIs



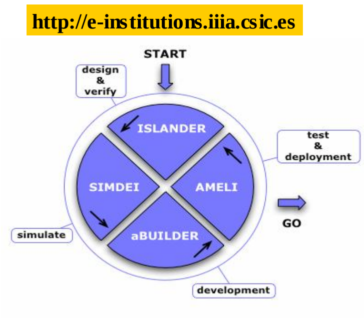
The **World of interest** is application-specific and refers to the part of the world that is relevant to the application. F.i. for a climate control application, the WoI comprises rooms, doors, heaters, etc.

Bridging Electronic Institutions with the WoI

- AMELI allows to incorporate entities in the world of interest by adding services.
- Examples of services:
  - TimeService – Management of timers.
  - NormService – Tracking of agents' normative positions.
  - ReputationService – Centralised reputation service.
  - ......

Overview

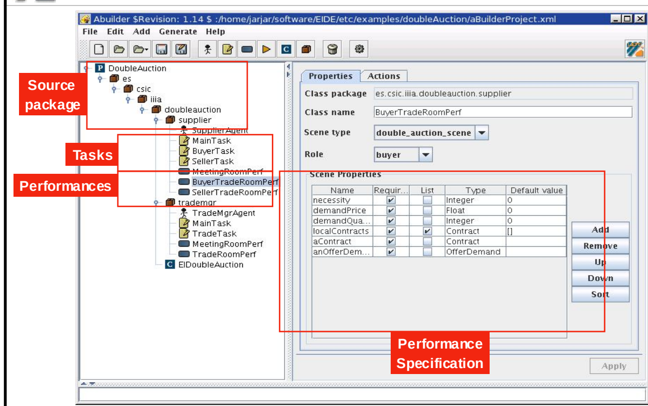
Electronic Institutions Development Environment



Agent Builder

- Goal: To ease agent development
- Approach:
  - Graphical specification of an agent's inner behaviour
  - Automatic generation of agent skeletons via graphical tools
  - Based on graphical specifications produced with ISLANDER
  - Agent architecture based on tasks and performances
    - Performance – Actions within a particular scene
    - Task – Sequence of performances related by a performative structure path

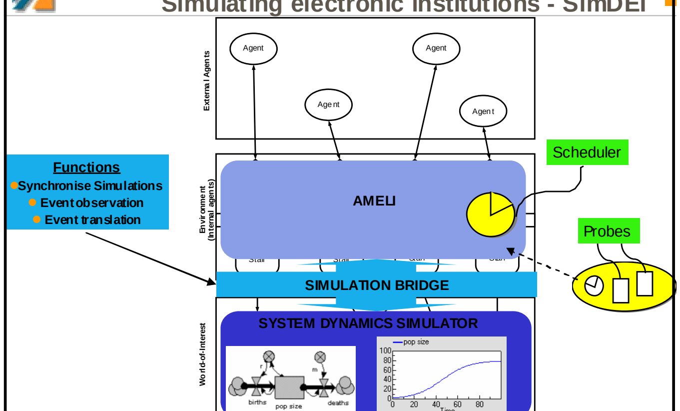
Agent Builder GUI



SimDei

- Goal: To verify dynamic properties of EIs
- Approach:
  - To run discrete event simulations at the object level
  - Simulations include interleaving of Eis with some simulator of the world-of-interest

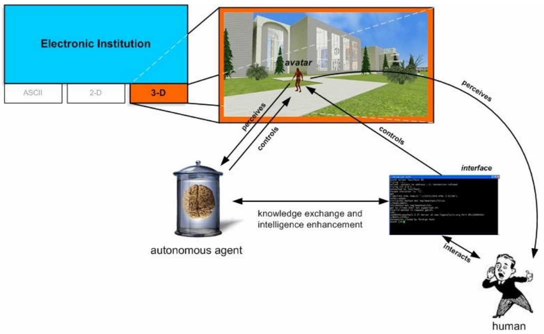
Simulating electronic institutions - SimDEI



Overview

I. Introduction
II. Specifying electronic institutions
III. Running electronic institutions
IV. Are Electronic Institutions Enough?
V. Electronic Institutions Development Environment
VI. 3D Electronic Institutions
VII. Conclusions

The Vision



Objectives

- New Metaphor: 3D Electronic Institutions
- Establish Trust
- Interact within a Social Environment
- Participants are
  - Human Users
  - Autonomous Software Agents

Trust

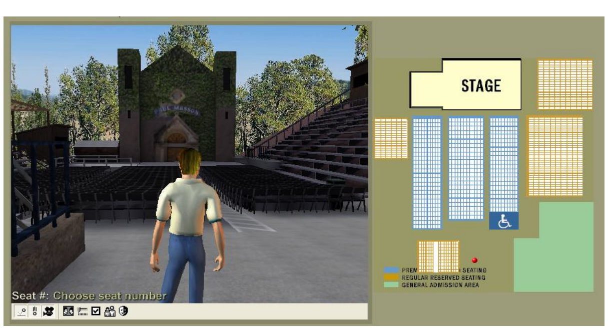
- One of the most important social concepts
- Helps humans to cope with their social environment
- It is present in all human interaction
- Implicit *trust assessment* underlies every traditional bargain
- Drawback in electronic markets: face-to-face interaction is missing
- Increased risk level related to online markets

Social Environments

- Social interaction is the key feature
- Conversations
- Virtual Worlds are spaces were people *meet*
- Strong relation to Trust

"Whom do you know?"

3D Virtual Worlds



3D Virtual Worlds

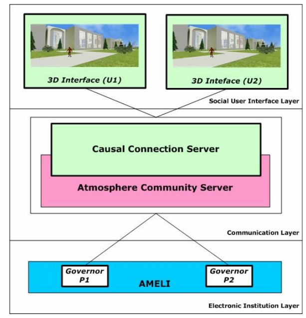
- Space is designed and arranged according to *human daily experiences*
- Space and objects in space produce an *immersive environment*
- Construct a *virtual representation* of a particular domain
- A convenient and alternative *interface* for user interaction
- Social experience: already through the simple presence of others
- Versatile ways of Interaction

3D Electronic Institutions

3D Electronic Institutions combine the two metaphors of *Electronic Institutions* and *3D Virtual Worlds* into one single metaphor

- retain the features and advantages of the original metaphors
- the *essence* is “opening“ Electronic Institutions to human users
- explore the relationship between humans and software agents in a 3D Virtual Space

3D Electronic Institution



Communication Layer

- Actions performed by users are passed in terms of *messages* from the user interface to the communication layer
- The Causal Connection Server captures these messages and postpones the actual execution
- Messages are sent to AMELI in order to determine their “validity“ and checks whether a particular message goes in line with the Electronic Institution rules or not
- A positive validation results in executing the requested action

Communication Layer

- Causal Connection: “whenever a change is made in the representation of the system, the system itself changes to maintain a consistent state and vice versa“
- Reflective systems: “the representation of the system is part of the system itself“

Social User Interface Layer

- 3D User Interface
- Adobe Atmosphere Player
  - Free download from Adobe Website
  - Player is embedded in HTML Page
  - Multimedia (Sound, Movies, Live Streams ...)
  - A combination of JavaScript and Java is used to communicate with “outer world“
- One Player per User

Social User Interface Objectives

- Awareness
  - knowledge about my own *location* and the location of other participants
  - being aware of different *types* of participants (human users or software agents)
  - distinguish between internal and external participants (avatar visualization code)

Social Virtual World

Implementation

- 3D Electronic Institution Specification
  - As an XML schema which enhances the Electronic Institution Specification
- Specification Transformer
  - Automatically converts the specification of an Electronic Institutions into a 3D Electronic Institution Specification

Annotation Editor

- Design Tool for System Engineers
- Input: Electronic Institution Specification
- Output: 3D Virtual World
- A basic conversion is done automatically
- Additional features are incorporated by the System Engineer (eg., furniture)

Conclusions

- Engineering open multi-agent systems is a highly complex task.
- Electronic institutions reduce this complexity by introducing normative (regulatory) environments.
- We have presented **an Electronic Institutions Development Environment (EIDE) that facilitates the deployment of electronic institutions**.
- EIDE targeted at supporting **environment engineering** in open multi-agent systems.
- 3D Electronic institutions as a user friendly environment

People

- IIIA Researchers: Carles Sierra, Juan Antonio Rodriguez-Aguilar, Josep Lluis Arcos, Pablo Noriega
- IIIA developers: Bruno Rosell, David de la Cruz, Guifre Cuni
- E-Markets Group: Anton Bogdanovych, Simeon Simoff, Helmut Berger