HETEROGENEOUS AGENT SYSTEMS

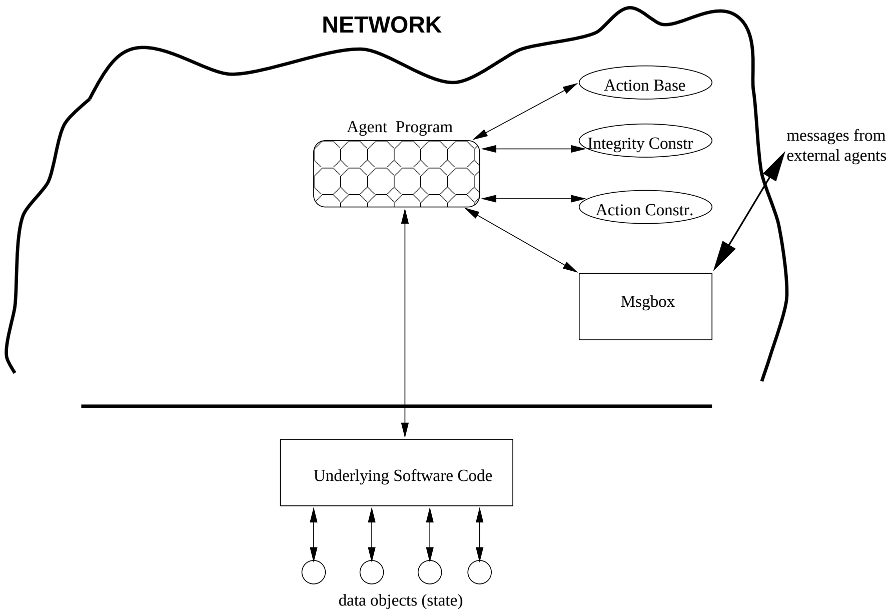
V.S.Subrahmanian
University of Maryland
:
(with P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Ozcan, R. Ross)

October 25, 2000

Recall

- Each agent is built on top of some codebase having some associated set of data structures (including a message box).
- Agent state at time t: whatever is on those data structures at time t.
- Whenever agent's state changes, agent must take "appropriate" action(s).

The goal of this section is to develop principled ways of coupling actions to agent state changes.



Agent Architecture

Action Base: Action

An *action* α consists of six components:

Name: $\alpha(\mathsf{X}_1, \ldots, \mathsf{X}_n)$, where the X_i's are root variables.

Schema: $(\tau_1, \ldots, \tau_n)$, of types.

Action Code: This is a body of code that executes the action.

Pre: A code-call condition χ, called the *precondition* of the action, denoted by $Pre(\alpha)$ ($\mathbf{pre}(\alpha)$ must be *safe modulo the variables* $\mathsf{X}_1,\ldots,\mathsf{X}_n$);

Add: a set $\mathbf{add}(\alpha)$ of code-call conditions;

Del: a set $\mathbf{del}(\alpha)$ of code-call conditions.

An *action atom* is a formula $\alpha(t_1, \ldots, t_n)$, where t_i is a term, i.e., an object or a variable, of type τ_i, for all $i = 1, \ldots, n$.

Actions: Differences with AI definition

Item	**Classical AI**	**Our framework**
Agent State	Set of logical atoms	Arbitrary data structures
Precondition	Logical formula	Code call condition
Add/delete list	set of ground atoms	Code call condition
Action Implementation	Via add list and delete list	Via arbitrary program code
Action Reasoning	Via add list and delete list	Via add list and delete list

Actions: Differences with AI definition

1. In AI, states are *physically modified* by union-ing the current state with items in the add list, and then deleting items in the delete list.
2. In our framework, the add-list and the delete-list do not physically implement the action.
3. They are used by the agent for reasoning.

In-class exercise

Recall the bookstore with a relational **`inventory`** table. How would you describe an action called **`update_qty`**?

In-class exercise

Recall the bookstore with a relational **inventory** table. How would you describe an action called **add_to_inventory**?

$\boxed{\text{Action Base}}$

A finite set of actions.

(θ, γ)-Executability

- Let $\alpha(\vec{X})$ be an action.
- Let $\mathcal{O}$ be the current state of some agent.
- A ground instance $\alpha(\vec{X})\theta$ of $\alpha(\vec{X})$ is *executable* in state $\mathcal{O}_{\mathcal{S}}$ iff there exists a solution γ of $Pre(\alpha(\vec{X}))\theta$ w.r.t. $\mathcal{O}$.
- In this case, $(\alpha(\vec{X}), \theta, \gamma)$ is a feasible *execution triple* for $\mathcal{O}$.
- $\Theta\Gamma(\alpha(\vec{X}), \mathcal{O})$ denotes the set of all pairs (θ, γ) such that $(\alpha(\vec{X}), \theta, \gamma)$ is a feasible execution triple in state $\mathcal{O}$.

Action execution

- Suppose $(\alpha(\vec{X}), \theta, \gamma)$ is a feasible execution triple in state $\mathcal{O}$.
- The state which results upon (θ, γ)-execution of $\alpha(\vec{X})$ satisfies all code call conditions that $\mathcal{O}$ satisfied except that:
  - code call conditions in $\mathbf{add}(\alpha(\vec{X}))\theta\gamma$ are true in the new state and
  - code call conditions in $\mathbf{del}(\alpha(\vec{X}))\theta\gamma$ are false in the new state.

Concurrent action execution: in-class discussion

What does it mean to execute 2 actions simultaneously?

Many possible answers.

Notion of Concurrency

A *notion of concurrency* is a function, conc such that:

- Its input is an object state $\mathcal{O}$ and a set AS of execution triples;
- Its output is a single execution triple such that:
  - if $AS = \{\alpha\}$ is a singleton action, then $\mathsf{conc}(\mathcal{O}, AS) = \alpha$.
  - if $AS_1 \subseteq AS_2$ and $\mathsf{conc}(\mathcal{O}, AS_i) = (\alpha_i(\vec{X}_i), \theta_i, \gamma_i)$ for $i = 1, 2$, and α_2 is (θ_2, γ_2)-executable in state $\mathcal{O}$, then α_1 is (θ_2, γ_2) executable in state $\mathcal{O}$.

What are the pros and cons of the above definition? Should we drop the second axiom? Keep it as is? Class discussion.

Weakly-Concurrent Execution (WCE)

- Suppose AS is a set of feasible execution triples in state $\mathcal{O}$.
- The weakly concurrent execution of AS yields a state $\mathcal{O}'$ just like $\mathcal{O}$ except that:
  - Every code call condition in the delete list of any feasible execution triple of AS is falsified and
  - Every code call condition in the add list is *later* made true.

Additions are done after deletions.

Example of WCE

- Consider the actions `update_qty` and `add_to_inventory`.
- Suppose we use WCE to concurrently execute `update_qty(‘‘Topaz’’,20)` and `update_qty(‘‘Virus,33)`. What happens?
- Suppose we use WCE to concurrently execute `update_qty(‘‘The Rope’’,25)` and `add_to_inventory(<‘‘The Rope’,...,>)`. What happens?
- Suppose we use WCE to concurrently execute `update_qty(‘‘Topaz’’,20)` and `update_qty(‘‘Topaz’’,30)`. What happens?

In-class discussion: Problems with WCE

Sequential Concurrent Execution

Suppose $AS = (\alpha_i(\vec{X}_i, \theta_i, \gamma_i)) \mid 1 \leq i \leq n\}$ is a set of feasible execution triples. AS is said to be S-*concurrently executable* in state $\mathcal{O}$ iff there exists a permutation π of AS and a sequence of states $\mathcal{O}^0, \ldots, \mathcal{O}^n$ such that:

- $\mathcal{O} = \mathcal{O}^0$;
- for each $1 \leq i \leq n$, $\alpha_i(\vec{X}_i)$ us (θ_i, γ_i)-executable in state $\mathcal{O}_{i-1}$ and
- the new state resulting from the (θ_i, γ_i)-execution of $\alpha_i(\vec{X}_i)$ in state $\mathcal{O}_{i-1}$ is $\mathcal{O}_i$.

In-class discussion: Example

- Consider the actions `update_qty` and `add_to_inventory`.
- Suppose we use SCE to concurrently execute `update_qty(``Topaz'',20)` and `update_qty(``Virus,33)`. What happens?
- Suppose we use SCE to concurrently execute `update_qty(``The Rope'',25)` and `add_to_inventory(<``The Rope',...,>)`. What happens?
- Suppose we use SCE to concurrently execute `update_qty(``Topaz'',20)` and `update_qty(``Topaz'',30)`. What happens?

In-class discussion: SCE vs. WCE

Full-Concurrent Execution (FCE)

- Every permutation of the actions in AS must be executable and
- Each such permutation must lead to the same final state.

In-class discussion: Example

- Consider the actions `update_qty` and `add_to_inventory`.
- Suppose we use FCE to concurrently execute `update_qty(‘‘Topaz’’,20)` and `update_qty(‘‘Virus,33)`. What happens?
- Suppose we use FCE to concurrently execute `update_qty(‘‘The Rope’’,25)` and `add_to_inventory(<‘‘The Rope’,...,>)`. What happens?
- Suppose we use FCE to concurrently execute `update_qty(‘‘Topaz’’,20)` and `update_qty(‘‘Topaz’’,30)`. What happens?

In-class discussion: FCE vs. WCE and SCE

Complexity results

Theorem.
Given as input, a set AS of execution triples feasible wrt an input state $\mathcal{O}$, checking if AS is S-concurrently executable is NP-complete

Proof - membership in NP

- Guess a permutation.
- Check if works !
- Latter step is polynomially evaluable (why?).

Proof - NP-hardness idea

- Reduction from M3SAT.
- SAT: Suppose $\phi = \wedge_{i=1}^{m} C_i$ where $C_i = L_{i,1} \vee \cdots \vee L_{i,k_i}$ is a disjunction over propositional symbols $X = \{x_1, \ldots, x_n\}$.
- 3SAT requires each $k_i \leq 3$.
- M3SAT (Monotone 3SAT) requires all literals in any C_i to be either positive or negative.
- Reduce M3SAT instances to our problem.
- Read detailed proof in book.

Complexity results

Theorem.
Given as input, a set AS of execution triples feasible wrt an input state $\mathcal{O}$, checking if AS is F-concurrently executable is coNP-complete.

Proof - membership in coNP

- Guess 2 permutations.
- If one of them isn't feasible or if they lead to different results, AS is not F-concurrently executable.
- Latter step is polynomially evaluable (by same reasoning as for S-concurrent execution).

For hardness, reduce complement of M3SAT.

Agent programs

- An agent program is a set of rules.
- A rule has the form $Head \leftarrow Body$ (on a computer, $\leftarrow$ is replaced by $:-$).
- To describe what $Head$ and $Body$ may look like, we need to define *status atoms*.

Status Atoms

- *Action atom* is of the form $\alpha(t_1, \ldots, t_n)$, where t_i is a term, i.e., an object or a variable, of type τ_i, for all $i = 1, \ldots, n$.
- *Modalities:*
  - **P** - permitted
  - **F** - forbidden
  - **O** - obligatory
  - **W** - waived
  - **Do** - do
- If Op is a modality and $\alpha(t_1, \ldots, t_n)$ is an action atom, then $Op\alpha(t_1, \ldots, t_n)$ is a *Status Atom.*
- If A is a status atom, then A and $\neg A$ are *Status Literals.*

Action Rule

An *action rule* r has the form

$$A \leftarrow \chi \& L_1 \& \ldots \& L_n$$

where A is a status atom, χ is a code call condition, and each of $L_1, \ldots, L_n$ are status literals.
Informally: if the current agent state satisfies χ and $L_1, \ldots, L_n$ hold, then A holds as well.

In-class exercise

Suppose you have an action called `order_book(Supplier,Title,Num)`. Write a rule on top of our `inventory` database that orders 5 copies of book whenever its Qty is below its Rstk level.

DO order_book(X.Sup,X.Title,5) $\leftarrow$
in(X,db:sql(**"select * from inventory where qty < rstk"**)) &

In-class exercise

Suppose you have a second database called `order_amount` which for each book, specifies how much of the book to order when an order is placed. Suppose this relation has fields "Title" and "Amount." Extend the previous example so that the amount of the book ordered is extracted from this relation.

DO order_book(X.Sup,X.Title,Y.amount)) $\leftarrow$
in(X,db:sql(**"select * from inventory where qty < rstk"**)) &
in(Y,db:sql(**"select * from order_amount where Title = X.Title"**)).

In-class exercise

Suppose you have a third relation called `onorder` which specifies if a book has been ordered. Expand the previous example so that a book is ordered only if the previous conditions are satisfied and the book is not on order.

DO order_book(X.Sup,X.Title,Y.amount)) ←
in(X,db:sql(“**select * from inventory where qty < rstk**”)) &
in(Y,db:sql(“**select * from order_amount where Title = X.Title**”)) &
is({},db:sql(“**select * from onorder where Title = X.Title**”)).

$\boxed{\text{In-class exercise}}$ Alternatively, write two rules. Order_book is forbidden if the book has been previously ordered. Order the book if ordering it is not forbidden and its quantity is below the restock amount. Modify what you wrote earlier.

Two communicating agents

- Agent A: has a data structure called "todo" and a code call called "check" If there's anything returned, it sends a message containing one of the items returned to agent B and deletes that item from todo. For this, it has a function called delete.
- Agent B: has a message box. Whenever anything from A arrives in the message box, it sends A and acknowledgement and then deletes the message from the message box.

What are the data structures, functions, etc. used by agents A and B? What kinds of rules do they use?

Semantics of agents

- Suppose we have an agent **aga** in state $\mathcal{O}$.
- Remember that **aga** has an associated:
  - set of data structures
  - set of API function calls
  - set of actions
  - agent program
  - set of integrity constraints
  - notion of concurrency
  - set of action constraints
- What is the set of actions that the agent must concurrently execute?
- To do this, we define the notion of a *feasible status set* containing ground status atoms.

- The set of actions that the agent must concurrently execute is whatever is in a feasible status set.

Feasible Status Sets

- A *status set* is a finite set of ground status atoms.
- Status set S is feasible for agent **aga** in state $\mathcal{O}$ if it satisfies the following:
  - Deontic and Action Consistency
  - Deontic and Action Closure
  - Closure under Program Rules
  - State Consistency

We study these one by one.

Deontic Consistency

S is deontically consistent w.r.t. state $\mathcal{O}$ iff:

- If $\mathbf{O}\alpha \in S$, then $\mathbf{W}\alpha \notin S$
- If $\mathbf{P}\alpha \in S$, then $\mathbf{F}\alpha \notin S$
- If $\mathbf{P}\alpha \in S$, then $(\exists)pre(\alpha)$ is true in $\mathcal{O}$.

Action Consistency

- S is *action consistent* iff S satisfies all ground instances of all action constraints w.r.t. state $\mathcal{O}$.
- S satisfies a ground action constraint

$$\{\alpha_1, \ldots, \alpha_n\} \hookleftarrow \chi$$

iff either $\mathcal{O}$ does not satisfy χ or $\{\alpha_1, \ldots, \alpha_n\} \not\subseteq \{\alpha \mid \mathbf{Do}\alpha \in S\}$.

Deontic and Action Closure

The *deontic closure*, dcl(S), of a status set S is the closure of S under the rule `If` $\mathbf{O}\alpha \in S$`, then` $\mathbf{P}\alpha \in S$.
The *action closure*, acl(S), of a status set S is the closure of S under the rules:

- If $\mathbf{O}\alpha \in S$, then $\mathbf{Do}\alpha \in S$,
- If $\mathbf{Do}\alpha \in S$, then $\mathbf{P}\alpha \in S$,

S is action closed iff $acl(S) = S$. Similarly for deontically closed.

$\boxed{\text{Theorems}}$

Suppose S is a status set. Then,

1. $acl(S) = S$ implies $dcl(S) = S$.
2. $dcl(S) \subseteq acl(S)$ for all S.

Proofs

Closure under program rules: operator App

$App(S) = \{A \mid A \leftarrow \chi \,\&\, L_1, \ldots \&\, L_n$ is a ground instance of a rule r in the agent's agent program and:

- χ is true in state $\mathcal{O}$ and
- $B^{+}_{as}(r) \subseteq S$ and
- $\neg.B^{-}_{as}(r) \cap S = \emptyset$ and
- for every atom $Op(\alpha) \in B^{+}(r) \cup \{A\}$ such that $Op \in \{\mathbf{P}, \mathbf{O}, \mathbf{Do}\}$, the action α is executable in state $\mathcal{O}$.

S is closed under **aga**'s rules iff $App(S) \subseteq S$.

Feasible status set

Suppose **aga** is an agent, $\mathcal{O}$ its current state and S is a status set. S is feasible iff:

- $App \subseteq S$;
- S is deontically and action consistent;
- S is action closed and deontically closed;
- $\mathcal{O}'$ satisfies all of **aga**'s integrity constraints where $\mathcal{O}'$ is the result of executing action $\mathsf{conc}(\mathbf{Do}(S))$ in state $\mathcal{O}$ — $Op(S) = \{\alpha \mid Op\alpha \in S\}$.

Example State: inventory

Lname	Fname	Title	Qty	Rstk	Sup	Buy	Sell
John	Keats	Odes	15	5	ABC Books	20	38
Leon	Uris	Topaz	10	3	XYZ Books	6	11
Robin	Cook	Virus	20	6	A1 Books	4	14
...	...	...	...	...	...	...	...

Example State: order_amount

Title	Amout
Odes	5
Topaz	10
Virus	25
...	...

Example State: onorder

Virus

Example rule set

DO order_book(X.Sup,X.Title,Y.amount)) $\leftarrow$
in(X,db:sql(“**select * from inventory where qty < rstk**”)) &
in(Y,db:sql(“**select * from order_amount where Title = X.Title**”)) &
is({},db:sql(“**select * from onorder where Title = X.Title**”)).

What is a feasible status set?

Another example rule set

DO order_book(X.Sup,X.Title,Y.amount)) ←
in(X,db:sql(**“select * from inventory where qty < rstk”**)) &
in(Y,db:sql(**“select * from order_amount where Title = X.Title”**)) &
is({},db:sql(**“select * from onorder where Title = X.Title”**)).
F order_book(X.Sup,X.Title,Y.amount)) ←
in(X,db:sql(**“select * from inventory where qty < rstk”**)) &
X.Buy ¿ 25.

What is a feasible status set?

Another example rule set

DO order_book(X.Sup,X.Title,Y.amount)) ←
in(X,db:sql(“**select * from inventory where qty < rstk**”)) &
in(Y,db:sql(“**select * from order_amount where Title = X.Title**”)) &
is({},db:sql(“**select * from onorder where Title = X.Title**”)).
F order_book(X.Sup,X.Title,Y.amount)) ←
in(X,db:sql(“**select * from inventory where qty < rstk**”)) &
X.Buy ¿ 10.

What is a feasible status set?

$\boxed{\text{Another example}}$

$$\mathbf{O}\alpha \leftarrow .$$

Find a feasible status set?

$\boxed{\text{Another example}}$

$$\mathbf{Do}\alpha \leftarrow .$$

Find a feasible status set?

$\boxed{\text{Another example}}$

$$\mathbf{Do}\alpha \leftarrow \neg\mathbf{P}\beta$$

Find a feasible status set?

Another example

$$\mathbf{Do}\alpha \leftarrow \neg\mathbf{P}\beta$$
$$\mathbf{Do}\beta \leftarrow \neg\mathbf{P}\alpha$$

Find a feasible status set?

Another example

$$\mathbf{Do}\alpha \leftarrow \neg\mathbf{Do}(\alpha).$$

Find a feasible status set?

Another example

$$\mathbf{Do}\alpha \leftarrow$$

- Current state $\mathcal{O}$ says $in(c, d : f(a))$ is true, but $in(X, d : f(a))$ is false for all $X \neq c$.
- $Add(\alpha)$ says $\{in(b, d : f(a)), in(c, d : f(a))\}$ Agent **aga** has an integrity constraint
$$in(c, d : f(a)) \leftarrow$$
- $Del(\alpha) = \{in(c, d : f(a))\}$
- $Pre(\alpha) = true.$
- Find a feasible status set?

Another example

$$\mathbf{Do}\alpha \leftarrow$$
$$\mathbf{F}\alpha \leftarrow$$

Find a feasible status set?

Example

What happens in the following case? Agent **aga** has:

1. One action α with $pre(\alpha) =$ "true," $add(\alpha) = \{in(a(d : f(b))\}$ and $del(\alpha) = \emptyset$.
2. One IC: $in(a(d : f(b)))$.
3. Current state $\mathcal{O}$ falsifies the above IC.
4. Agent program: $\mathbf{P}a \leftarrow$.
5. There is only one feasible status set, viz. $\{\mathbf{P}a, \mathbf{Do}a\}$.
6. $\{\mathbf{P}a\}$ is not feasible. Why?

$\boxed{\text{Theorem}}$

Let S be a feasible status set. Then,

1. If $\mathbf{Do}(\alpha) \in S$, then $\mathcal{O}$ satisfies $pre(\alpha)$;
2. If $\mathbf{P}\alpha \notin S$, then $\mathbf{Do}(\alpha) \notin S$;
3. If $\mathbf{O}\alpha \in S$, then $\mathcal{O}$ satisfies $pre(\alpha)$;
4. If $\mathbf{O}\alpha \in S$, then $\mathbf{F}\alpha \notin S$.

Proofs

Rational Status Sets

- Suppose we have an agent with no ICs and ACs, and a single action with a "true" precondition.
- Note that with a single agent rule of the form $\mathbf{P}a$, we may have two status sets: $\{\mathbf{P}a\}$ and $\{\mathbf{P}a, \mathbf{Do}a\}$.
- Suppose no other information is available.
- In this case, it may be argued that the first status set $\{\mathbf{P}a\}$ should be chosen instead of the second.
- However, feasible status sets do not distinguish between the two and an agent using feasible status set semantics can choose either of the two nondeterministically.
- The *rational status set* semantics tries to select "minimalistic" feasible status sets.

Groundedness

- A status set S is *grounded*, if there exists no status set $S' \subset S$ such that S' satisfies all conditions of the definition of a feasible status set (except the integrity constraint condition).
- S is a *rational status set* iff S is feasible and S is grounded.

Example

What happens in the following case? Agent **aga** has:

1. One action α with $pre(\alpha) =$ "true," $add(\alpha) = \{in(a(d : f(b))\}$ and $del(\alpha) = \emptyset$.
2. One IC: $in(a(d : f(b))$.
3. Current state $\mathcal{O}$ falsifies the above IC.
4. Agent program: $\mathbf{P}a \leftarrow$.
5. There is only one feasible status set, viz. $\{\mathbf{P}a, \mathbf{Do}a\}$.
6. $\{\mathbf{P}a\}$ is not feasible. Why?
7. But it has no rational status set !

Example

What happens when agent **aga** has the following agent program and we make different types of assumptions about its other components:

$$\begin{aligned} \mathbf{P}a &\leftarrow \neg\mathbf{P}b \\ \mathbf{P}b &\leftarrow \neg\mathbf{P}a \end{aligned}$$

If both a, b have preconditions true in the current state, and there are no ICs, then $\{\mathbf{P}a\}$ and $\{\mathbf{P}b\}$ are rational. But $\{\mathbf{P}a, \mathbf{Do}a\}$ is not.

Theorem

- An agent program rule is *positive* if no negated action status literal appears in its body, i.e. the rules body contains no action status literal of the form $\neg Op\alpha(\ldots)$.
- An agent program rule *positive* if all its rules are positive.
- **Theorem.** If an agent with a positive agent program has a rational status set w.r.t. some state $\mathcal{O}$, then that rational status set is unique w.r.t. $\mathcal{O}$.

$\mathcal{T}_{\mathcal{P}}$ Operator

Suppose $\mathcal{P}$ is an agent program and $\mathcal{O}$ an agent state. Then, for any status set S,

$$\mathcal{T}_{\mathcal{P}}(S) = App(S) \cup dcl(S) \cup acl(S).$$

Intuition:

- Close S under agent program rules;
- Close S w.r.t. deontic closure;
- Close S w.r.t. action closure.

Theorem

Suppose $\mathcal{P}$ is an agent program, $\mathcal{O}$ an agent state, and suppose S is any agent state. If S satisfies $(S1)$ and $(S3)$ of feasibility, then S is pre-fixpoint of $\mathcal{T}_{\mathcal{P}}$, i.e., $\mathcal{T}_{\mathcal{P}}(S) \subseteq S$.
(S1) is closure under program rules, and (S3) is deontic and action closure.

Fixed Points

- If $f : S \rightarrow S$ is a function, and $x \in S$ and $f(x) = x$, then x is called a *fixed point* of f.
- x is a *prefixpoint* of f iff $f(x) \leq f$.
- If $\leq$ is a partial ordering on S, a fixed-point x of f is called *minimal* iff there is no other fixed point $y < x$ (i.e. $y \leq x$ and $y \neq x$).
- x is a *prefixpoint* of f iff $f(x) \leq f$.
- If S is a complete lattice under $\leq$, then f has a unique minimal fixpoint, a unique minimal prefixpoint and the two coincide.

Theorem

1. $\mathcal{T}_{\mathcal{P}}$ is monotonic, i.e. $S_1 \subseteq S_2$ implies that $\mathcal{T}_{\mathcal{P}}(S_1) \subseteq \mathcal{T}_{\mathcal{P}}(S_2)$.
2. $\mathcal{T}_{\mathcal{P}}$ has a unique minimal fixed point, denoted $lfp(\mathcal{T}_{\mathcal{P}})$ when we consider the $\subseteq$ ordering on status sets.

$\boxed{\text{Theorem}}$

S is a rational status set for an agent **aga** iff $S = lfp(\mathcal{T}_{\mathcal{P}})$ and S is a feasible status set.

Proof ($\Rightarrow$)

- Suppose $S = lfp(\mathcal{T}_{\mathcal{P}})$ is a rational status set.
- Then S is feasible by definition of rational status set.
- So S is a pre-fixpoint of $\mathcal{T}_{\mathcal{P}}$.
- Hence, $lfp(\mathcal{T}_{\mathcal{P}}) \subseteq S$ because $lfp(\mathcal{T}_{\mathcal{P}})$ is a prefixpoint of $\mathcal{T}_{\mathcal{P}}$.
- Clearly, $lfp(\mathcal{T}_{\mathcal{P}})$ satisfies $(S1)$ and $(S3)$.
- Also, $lfp(\mathcal{T}_{\mathcal{P}})$ satisfies (S2) [deontic/action consistency].
- Hence, $lfp(\mathcal{T}_{\mathcal{P}}) = S$.

$\boxed{\text{Proof } (\Leftarrow)}$

- Suppose $S = lfp(\mathcal{T}_{\mathcal{P}})$ is a feasible status set.
- As every status set S' which satisfies $(S1)$–$(S3)$ is a pre-fixpoint of $\mathcal{T}_{\mathcal{P}}$ and $lfp(\mathcal{T}_{\mathcal{P}})$ is the least prefix point, $S' \subseteq S$ implies $S = S'$.
- It follows that S is rational.

Corollaries

1. Suppose agent **a** has a positive agent program. Then, for every state $\mathcal{O}$, **a** has at most one rational status set.
2. Under the above conditions: if no integrity constraints are present, then **a** has a rational status set iff it has a feasible status set.

Reasonable status sets

- Consider

$$\mathbf{Do}(\alpha) \leftarrow \neg\mathbf{Do}(\beta).$$

- When no ICs, ACs are involved and α's precondition is true in the current state, we have two rational status sets.
- $S_1 = \{\mathbf{Do}(\alpha), \mathbf{P}(\alpha)\}$
- $S_2 = \{\mathbf{Do}(\beta), \mathbf{P}(\beta)\}$
- The second may seem less intuitive than the first. Why?

Reasonable status sets attempt to give "directionality" to rules.

Agent program reduct

- Suppose we have a *ground* rule r and a status set S and an agent state $\mathcal{O}$.
- The *reduct* of rule r w.r.t. the above is a set $\Xi_{r,S}$ constructed as follows:
  - If r is positive, then $\Xi_{r,S} = \{r\}$;
  - Else
    - If $B^{-}_{as}(r) \cap S \neq \emptyset$ then $\Xi_{r,S} = \emptyset$
    - else $\Xi_{r,S} = \{r'\}$ where r' is exactly like r except that all negative status atoms in its body are deleted.

$\boxed{\text{Example}}$

$$\mathbf{Do}(\alpha) \leftarrow \neg\mathbf{Do}(\beta).$$

$\boxed{\text{Example}}$

$$\mathbf{Do}(\alpha) \leftarrow \neg\mathbf{Do}(\alpha).$$

$\boxed{\text{Example}}$

$$\mathbf{Do}(\alpha) \leftarrow \mathbf{Do}(\alpha).$$

Agent program reduct continued

- Suppose we have an agent program P and a status set S.
- The reduct of P w.r.t. S is the set $\cup_{r \in grd(P)} \Xi_{r,S}$ where $grd(P)$ denotes all ground instances of P.

Note

- Reduct of P w.r.t. any S is always positive.
- If P is fixed and we make S larger, then reduct of P gets smaller.
- I.e. if $red(S, P)$ denotes the reduct of P w.r.t. S, then

$$S_1 \subseteq S_2 \Rightarrow red(S_1, P) \supseteq red(S_2, P).$$

Reasonable status sets

- If P is a positive agent program, then S is a *reasonable status set* for P on $\mathcal{O}$, if and only if S is a rational status set for P on $\mathcal{O}$.
- Otherwise: S is a *reasonable status set* for P on $\mathcal{O}$, if and only if S is a reasonable staus set of $red(S, P)$.

$\boxed{\text{Example}}$

$$\mathbf{Do}(\alpha) \leftarrow \neg\mathbf{Do}(\beta).$$

$\boxed{\text{Example}}$

$$\mathbf{Do}(\alpha) \leftarrow \neg\mathbf{Do}(\alpha).$$

Example

$$\mathbf{Do}(\alpha) \leftarrow \mathbf{Do}(\alpha).$$

Reading for next class

Y. Shoham. *Agent Oriented Programming,*, Artificial Intelligence, 60(1), pp. 51-92, 1993.
Reprinted in: Readings in Agents, eds. M. Huhns and M. Singh, Morgan Kaufman, 1998.