Planning and Pragmatics

Luciana Benotti and Patrick Blackburn

TALARIS Team
INRIA Lorraine, France
firstname.lastname@loria.fr

Deduction in Semantics, Stuttgart
11 October 2007
Motivations

- Drawing together Blackburn and Boss’ work on inference in DRT based architecture with Benotti’s work on adding planning to a text adventure game.

- Inspirations:
  - Clark: Language as Action
  - Thomason, Stone and deVault: Enlightened Update theory
  - Beaver and Zeevat: Fine Structure of Accommodation

- The present talk reports work in progress. We first discuss the general theme that interest us (Blackburn), take a closer look at planning and Enlightened Update (Benotti) and then try to draw some conclusions (Blackburn).
Recurrent themes

- Actions versus Information (reformulation of Beaver and Zeevat explicit addition principle)
- Physical Actions versus Linguistic Actions
- Accommodation versus Enlightenment (presuppositions and implicature)
- Planning versus Abduction (and logic programming with negation)
- Declarative versus Procedural (implicit timing via structure)
Recurrent themes

• Actions versus Information (reformulation of Beaver and Zeevat explicit addition principle)
• Physical Actions versus Linguistic Actions (salads, break fluid, magic screwdrivers, and flat tyres)
Recurrent themes

- Actions versus Information (reformulation of Beaver and Zeevat explicit addition principle)
- Physical Actions versus Linguistic Actions (salads, break fluid, magic screwdrivers, and flat tyres)
- Accommodation versus Enlightenment (presuppositions and implicature)
Recurrent themes

- Actions versus Information (reformulation of Beaver and Zeevat explicit addition principle)
- Physical Actions versus Linguistic Actions (salads, break fluid, magic screwdrivers, and flat tyres)
- Accommodation versus Enlightenment (presuppositions and implicature)
- Planning versus Abduction (and logic programming with negation)
Recurrent themes

- Actions versus Information (reformulation of Beaver and Zeevat explicit addition principle)
- Physical Actions versus Linguistic Actions (salads, break fluid, magic screwdrivers, and flat tyres)
- Accommodation versus Enlightenment (presuppositions and implicature)
- Planning versus Abduction (and logic programming with negation)
- Declarative versus Procedural (implicit timing via structure)
Perrault and others: Generation as Planning


- The theory of language is part of a general theory of action.
- Utterances are instances of speech acts.


- Speech acts are actions with preconditions and effects.
- The speaker has a goal and a plan for achieving the goal, and all speech acts are part of the plan.
- Language generation is modelled as planning.

Planning is the task of inferring a plan using a given initial state and a goal.
Plan recognition is the task of inferring a plan and a goal using observed actions.

- The speaker’s plan is inferred and refined observing his actions (his speech acts).
- Understanding an utterance involves understanding how this utterance plays a role in the speaker’s plan.
- This models accounts for direct and indirect speech acts and sentence fragments uniformly.
- This model explains the mechanisms that give rise to helpful responses.

A plan-based approach to speech act recognition. *Allen. 1979*

Discourse processing and commonsense plans. *Litman and Allen. 1990*
Other recent planning based work ...

- Sentence generation as planning. Koller and Stone. 2007
Obstacle elimination consists of:

1. recognising the plan of the interlocutor;
2. detecting obstacles to the plan as false preconditions;
3. adopting the goal of making these preconditions true;
4. forming a plan to carry this out; and
5. acting on this plan

- Accommodation is a special case of obstacle elimination.
- Opening a door for someone is a form of obstacle elimination. So is adding p to the presumptions when someone says something that presupposes p.

Accommodation, meaning, and implicature: Interdisciplinary foundations for pragmatics. Thomason. 1990
Thomason, Hobbs and others: Interpretation and Generation as Abduction

Interpretation as Abduction. *Hobbs, Stickel, Appelt, Martin. 1993*

- Recognising the speaker’s plan is a problem of abduction. Hence, interpretation is a problem of abduction.
- Planning is a type of abduction. Hence, generation is a problem of abduction.

Interrelating interpretation and generation in an abductive framework. *Thomason and Hobbs. 1997*

- An abductive approach to generation and interpretation reveals important commonalities between the two reasoning processes.

- Communicative intentions have preconditions, or ways that the world is assumed to be, and on which the success of the intention depends.
- It is these preconditions that we identify with speaker presuppositions, or the presuppositions of an utterance.
- Abduction is used as a unified architecture for the implementation of interpretation and generation.
Von Fintel: What is presupposition accommodation, again?

Attempts to clarify the role of accommodation and presupposition. Follows Stalnaker on insisting in the importance of timing.

Suppose Phoebe says I saw an interesting movie last night. To determine the content of her remark, one needs to know who is speaking, and so Phoebe, if she is speaking appropriately, must be presuming that the information that she is speaking is available to her audience ... that is shared information. But she need not presume that the information was available before she began to speak. The prior context that is relevant to the interpretation of a speech act is the context as it is changed by the fact that the speech act was made but prior to the acceptance or rejection of the speech act.
Implementing and extending the Enlightened Update theory
What is Enlightened Update?

Enlightened Update (EU) theory studies how common ground is constructed and exploited in coordinated activities (collaborative physical tasks and cooperative conversation).

General Claim: Coordinated activities move forward through:
- The \textbf{effects} of an action (or utterance), but also through
- The \textbf{assumptions} about the state of the world on which the success of the action depends

Enlightened Update and Tacit Action

It is mutual information that:

- C is a precondition for the action A, and
- An actor S executes A

⇒ S believes C

The authors of the theory argue that these tacit actions are intended by S when performing A.
Enlightened Update and Tacit Action

It is mutual information that:

- C is a precondition for the action A, and
- An actor S executes A

leads to S believes C

- C is a precondition for the action A, and
- An actor S executes A
- C is false

leads to
Enlightened Update and Tacit Action

It is mutual information that:

- C is a precondition for the action A, and
- An actor S executes A

⇒ S believes C

- C is a precondition for the action A, and
- An actor S executes A
- C is false

⇒ Tacit actions are executed to make C true (if possible)
Enlightened Update and Tacit Action

It is mutual information that:

• C is a precondition for the action A, and
• An actor S executes A

⇒ S believes C

• C is a precondition for the action A, and
• An actor S executes A
• C is false

⇒ Tacit actions are executed to make C true (if possible)

The authors of the theory argue that these tacit actions are intended by S when performing A
A New Laboratory for the Theory

- The theory of Enlightened Update was implemented and tested in the conversational agent COREF.
- In COREF, both dialogue participants are assumed to have the same background knowledge.
- We implement and test the theory when one of the participants has incomplete background knowledge.
- We implement enlightened update using planning rather than abduction.
- Our laboratory will be a text adventure game.

The New Laboratory
FrOzA: A Text-Adventure Game

FrOzA is a *text adventure* game:

1. FrOzA starts by describing the player location in the game world
2. The player types instructions that are interpreted and executed by the game
3. After executing an instruction, FrOzA describes its effects to the player

FrOzA is an extension of the text adventure engine FrOz developed by Koller, Debusmann, Gabsdil, and Striegnitz. 2004.
Inference tools are used to query and modify a game scenario.

- During action handling, the prover Racer and the planner Blackbox collaborate in order to find tacit actions
A Game Scenario: The Actions

- The action database includes all the actions that can be executed in a game scenario.
- The actions are represented in a STRIPS format.

<table>
<thead>
<tr>
<th>action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>take(patient:X)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>preconditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessible(X)</td>
</tr>
<tr>
<td>takeable(X)</td>
</tr>
<tr>
<td>not(has-loc(X player1))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>effects:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(delete) ( \exists Y. \text{has-loc}(X Y) )</td>
</tr>
<tr>
<td>(add) has-loc(X player1)</td>
</tr>
</tbody>
</table>

- Actions and preconditions are a natural way of thinking about enlightened update.
A Game Scenario: The Knowledge Bases

The world KB contains complete information about the world state

The world state

player(player1)
couch(couch1)
brown(couch1)
has-loc(player1,couch1)
apple(apple1)
has-loc(apple1,couch1)
drawing-room(room1)
has-loc(couch1,room1)
accessible(apple1)
takeable(apple1)
room(room2)
dragon(dragon1)
has-loc(dragon1,room2)

The player beliefs

player(player1)
A Game Scenario: The Knowledge Bases

The world KB contains complete information about the world state

The world state

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1, couch1)
- apple(apple1)
- has-loc(apple1, couch1)
- drawing-room(room1)
- has-loc(couch1, room1)
- accessible(apple1)
- takeable(apple1)
- room(room2)
- dragon(dragon1)
- has-loc(dragon1, room2)

The player beliefs

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1, couch1)
- apple(apple1)
- has-loc(apple1, couch1)
- drawing-room(room1)
- has-loc(couch1, room1)
A Game Scenario: The Knowledge Bases

The world KB contains **complete information** about the world state

### The world state

- `player(player1)`
- `couch(couch1)`
- `brown(couch1)`
- `has-loc(player1,couch1)`
- `apple(apple1)`
- `has-loc(apple1,couch1)`
- `drawing-room(room1)`
- `has-loc(couch1,room1)`
- `accessible(apple1)`
- `takeable(apple1)`
- `room(room2)`
- `dragon(dragon1)`
- `has-loc(dragon1,room2)`

### The player beliefs

- `player(player1)`
- `couch(couch1)`
- `brown(couch1)`
- `has-loc(player1,couch1)`
- `apple(apple1)`
- `has-loc(apple1,couch1)`
- `drawing-room(room1)`
- `has-loc(couch1,room1)`
- `accessible(apple1)`
- `takeable(apple1)`

You are in a brown couch.

There is an apple in the couch.

The couch is in a drawing-room.
A Game Scenario: The Knowledge Bases

The world KB contains **complete information** about the world state

---

**The world state**

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1, couch1)
- apple(apple1)
- has-loc(apple1, couch1)
- drawing-room(room1)
- has-loc(couch1, room1)
- accessible(apple1)
- takeable(apple1)
- room(room2)
- dragon(dragon1)
- has-loc(dragon1, room2)

---

**The player beliefs**

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1, couch1)
- apple(apple1)
- has-loc(apple1, couch1)
- drawing-room(room1)
- has-loc(couch1, room1)
- accessible(apple1)
- takeable(apple1)

---

You are in a brown couch.

There is a apple in the couch.

The couch is in a drawing-room.

> `take the apple`
A Game Scenario: The Knowledge Bases

The world KB contains complete information about the world state

The world state

player(player1)
couch(couch1)
brown(couch1)
has-loc(player1,couch1)
apple(apple1)
has-loc(apple1,couch1)
drawing-room(room1)
has-loc(couch1,room1)
ablessible(apple1)
takeable(apple1)
room(room2)
dragon(dragon1)
has-loc(dragon1,room2)

The player beliefs

player(player1)
couch(couch1)
brown(couch1)
has-loc(player1,couch1)
apple(apple1)
has-loc(apple1,couch1)
drawing-room(room1)
has-loc(couch1,room1)
ablessible(apple1)
takeable(apple1)
A Game Scenario: The Knowledge Bases

The world KB contains complete information about the world state

The world state

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1,couch1)
- apple(apple1)
- drawing-room(room1)
- has-loc(couch1,room1)
- accessible(apple1)
- takeable(apple1)
- room(room2)
- dragon(dragon1)
- has-loc(dragon1,room2)

The player beliefs

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1,couch1)
- apple(apple1)
- drawing-room(room1)
- has-loc(couch1,room1)
- accessible(apple1)
- takeable(apple1)

You are in a brown couch.
There is a apple in the couch.
The couch is in a drawing-room.
> take the apple
A Game Scenario: The Knowledge Bases

The world KB contains complete information about the world state

---

**The world state**

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1,couch1)
- apple(apple1)
- has-loc(apple1,player1)
- drawing-room(room1)
- has-loc(couch1,room1)
- accessible(apple1)
- takeable(apple1)
- room(room2)
- dragon(dragon1)
- has-loc(dragon1,room2)

---

**The player beliefs**

- player(player1)
- couch(couch1)
- brown(couch1)
- has-loc(player1,couch1)
- apple(apple1)
- has-loc(apple1,player1)
- drawing-room(room1)
- has-loc(couch1,room1)
- accessible(apple1)
- takeable(apple1)

---

You are in a brown couch.

There is an apple in the couch.

The couch is in a drawing-room.

> take the apple

You have the apple.
A Game Scenario: The Knowledge Bases

The world KB contains **complete information** about the world state

<table>
<thead>
<tr>
<th>The world state</th>
<th>The player beliefs</th>
</tr>
</thead>
<tbody>
<tr>
<td>player(player1)</td>
<td>player(player1)</td>
</tr>
<tr>
<td>couch(couch1)</td>
<td>couch(couch1)</td>
</tr>
<tr>
<td>brown(couch1)</td>
<td>brown(couch1)</td>
</tr>
<tr>
<td>has-loc(player1,couch1)</td>
<td>has-loc(player1,couch1)</td>
</tr>
<tr>
<td>apple(apple1)</td>
<td>apple(apple1)</td>
</tr>
<tr>
<td>has-loc(apple1,player1)</td>
<td>has-loc(apple1,player1)</td>
</tr>
<tr>
<td>drawing-room(room1)</td>
<td>drawing-room(room1)</td>
</tr>
<tr>
<td>has-loc(couch1,room1)</td>
<td>has-loc(couch1,room1)</td>
</tr>
<tr>
<td>accessible(apple1)</td>
<td>accessible(apple1)</td>
</tr>
<tr>
<td>takeable(apple1)</td>
<td>takeable(apple1)</td>
</tr>
<tr>
<td>room(room2)</td>
<td>room(room2)</td>
</tr>
<tr>
<td>dragon(dragon1)</td>
<td>dragon(dragon1)</td>
</tr>
<tr>
<td>has-loc(dragon1,room2)</td>
<td>has-loc(dragon1,room2)</td>
</tr>
</tbody>
</table>

We assume that the player KB contains:

- **incomplete knowledge** about the world state
- **mutual information** (shared by player and game) about **player beliefs**
Implementing and Testing the Theory
Let us see how is life without tacit actions

Imagine that you are giving instructions not to a computer game but to a person

The person answers as in the screenshot

Don’t you find this annoying? This system is not cooperating
Life WITH Tacit Action (A Cooperative Game)

- A cooperative game would do whatever is needed in order to execute the player instruction.
- The actions executed:
  1. (tacit) stand up
  2. (tacit) take the key
  3. (public) unlock the chest

In FrOzA, the prover Racer and the planner interact to:
- find a sequence of tacit actions and
- perform an enlightened update to the game scenario.
What information should be used in order to find the tacit actions?

1. The \textbf{complete information} about the world state stored in the world KB?

2. The \textbf{incomplete knowledge} about the world state stored in the player KB?

Let’s consider both possibilities in turn ...
Suppose that tacit actions are found using the complete information about the world state stored in the world KB.

- The actions executed are:
  1. (tacit) take the golden key
  2. (tacit) unlock the chest with the golden key
  3. (public) open the chest

- But the player doesn’t know which key unlocks the chest!

- To unlock the chest with the golden key cannot be part of the intention of the player.

- Hence, let’s consider the player knowledge base ...
Suppose that tacit actions are found using the incomplete information about the world state stored in the player KB.

- For this example no plan is found
- and this is just what we wanted.
- The player just does not have enough knowledge to leave this action tacit

But, what if ...
Tacit Action and Incomplete Knowledge 2

What if the golden key was stolen by a thief (without the player knowing) and the player says ‘Unlock the chest with the golden key’?

- In the player KB the key is accessible (it is on the table), then FrOzA decides to execute:
  1. (tacit) take the golden key
  2. (public) unlock the chest with the golden key

- But, this sequence of actions is not executable in the world because the key is no longer accessible (the thief has it)

We need both KBs:

- Tacit actions are inferred using the player KB
- Their execution is verified using the world KB
And there is more!

Incomplete Knowledge Leads to Learning ...
Incomplete Knowledge leads to Learning ...

Consider the following situation:

- The player doesn’t know which key unlocks the chest until she tries to unlock it with the right key.
- After the action succeeds, the player knows (and the game knows that the player knows) which key fits into the chest.

Remember: If C is a precondition for the action A, and S executes A publicly then S believes C.
Tacit Action and Learned Knowledge

Later in the game ...

The chest is again locked. **This time** the player just needs to make public her intention to open the chest.

- And the game infer the tacit action unlock:
  1. (tacit) take the golden key
  2. (tacit) unlock the chest with the golden key
  3. (public) open it

As knowledge grows, more actions can be left tacit.
Summary

- Enlightened update is an **indispensable ingredient** for a cooperative system.

- **We have viewed:**
  - Enlightened Update and Complete Knowledge
  - Enlightened Update and Incomplete Knowledge
  - Enlightened Update and Learned Knowledge

- **Tacit actions** should be inferred using **mutual information** about the beliefs of the actor, but their executability have to be verified wrt **complete information**.

- **Tacit actions** are **dynamic**: As the actor learns about the world, more actions can be left tacit.
How does this relate to the themes mentioned at the beginning?
The Principle of Explicit Addition: Accommodation is only possible in contexts where the explicit addition of the accommodated material would (i) produce a felicitous discourse, and (ii) result in a text which lacked the original presupposition.
Executing Tacit Actions

In FrOzA, after the tacit actions are found, the command that the player uttered is replaced by the command where the tacit actions are made explicit.

Example

Unlock the chest with the key is replaced by
Stand up, take the key and unlock the chest with the key

The tacit actions Stand up, take the key are explicitly added in order to eliminate the obstacles so that unlock the chest with the key can be successfully executed.
Tacit Action and Explicit Addition

Explicit Addition in FrOzA: Actions can be left tacit only in a state of the game where the explicit addition of the tacit actions would (i) succeed if executed in the current state of the game and (ii) result in an equivalent sequence of actions which lacked the the false preconditions.

The Principle of Explicit Addition: Accommodation is only possible in contexts where the explicit addition of the accommodated material would (i) produce a felicitous discourse, and (ii) result in a [equivalent] text which lacked the original presupposition.
Von Fintel: The Right Time

We can narrate our system reasoning in von Fintel terms:

The Right Time: The speaker is assuming that the common ground will satisfy the requirements by the time the update is to be performed. The speaker need not assume that the common ground prior to the utterance already has the right properties.

The Right Time in FrOzA: The player is assuming that the state of the game will satisfy the preconditions by the time the uttered command is to be executed.

It is between these two times \(\langle\text{time of utterance, time of execution}\rangle\) that tacit actions are performed.

When is pragmatic reasoning performed?

Recall Steenning’s discussion yesterday of the relative merits of pragmatic reasoning and logic programming with negation.

- Sometimes presented in first semantics then pragmatics fashion (e.g. Grice, Levinson).
- Already in Gazdar approach to presupposition, this is heavily modified.
- In van der Sandt model, largely abandoned.
Timing is important ...

• But timing is subtle.
• Usually not explicitly about time.
• Rather, it’s about how much structure has been built.
• Timing implicit — richer structures typically take more time to build.
Compatible with declarative approaches

- HPSG builds complex signs by combining different kinds of linguistic information.
- Similar strategy underlies the works of Hobbs and Enlightened Update. Complex representations are built that combines linguistic and non-linguistic information.

Such approaches are very suggestive as to what reasoning to an interpretation is about ...
Arguably complexity best handled in declarative approaches

- Work over theory on complex ontology.
- Build structure over this theory (Steenning is right, reasoning to an interpretation is about structure building).

Working in this way, Stenning’s sharp distinction between pragmatic reasoning and logic programming with negation seems to evaporate.
Structure building strategies

- Model building
- Abduction
- Planning
- Logic Programming of negation

All four approaches are intimately linked, certain versions of them may well be probably equivalent.
Summing up and reformulating:

- Rendering inferences
- Micro inferences (sub-sentential inferences)

Micro inference is essentially efficient structure building. Filtering rather than generate and test. Is this the essence of reasoning to an interpretation?
Further work

- Re-implementation of FrOzA
  - Original version in Oz which is no longer maintained
  - Generation and parsing are hardwired in the code
  - Currently implementing the kernel in Prolog, using off-the-shelf tools for specialised tasks (parsing, generation, planning, theorem proving).

- Intending to work with richer DRT based semantics

- Exploring relations between planning, abduction, model building and programming with negation.
Thanks for your attention!

Questions?