

# Domain-Level Reasoning for Dialogue Systems

Dirk Bühler

Institute of Information Technology,  
Ulm University

27 February 2009

# Outline

- ▶ Introduction
- ▶ Dialogue architecture for interactive reasoning
- ▶ Prototype systems
- ▶ Conclusions

# Outline

- ▶ Introduction
  - ▶ Spoken-language dialogue systems
  - ▶ Motivation and problem setting
- ▶ Dialogue architecture for interactive reasoning
- ▶ Prototype systems
- ▶ Conclusions

# Spoken-Language Dialogue Systems

- ▶ Goals: User-friendliness, efficiency, safety

# Spoken-Language Dialogue Systems

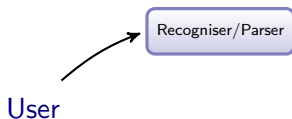
- ▶ Goals: User-friendliness, efficiency, safety

User

Travel info  
Navigation  
Calendar  
Infotainment  
...

# Spoken-Language Dialogue Systems

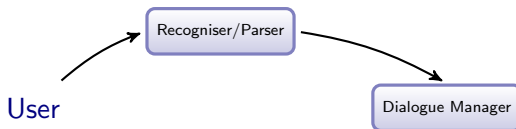
- ▶ Goals: User-friendliness, efficiency, safety



Travel info  
Navigation  
Calendar  
Infotainment  
...

# Spoken-Language Dialogue Systems

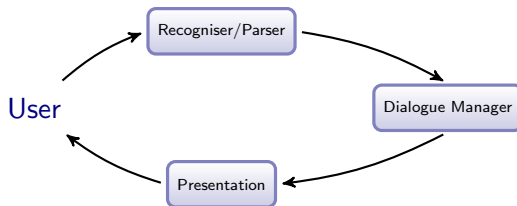
- ▶ Goals: User-friendliness, efficiency, safety



Travel info  
Navigation  
Calendar  
Infotainment  
...

# Spoken-Language Dialogue Systems

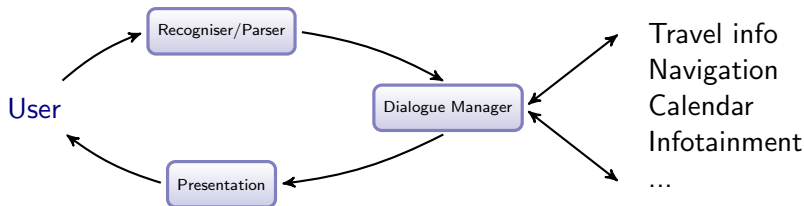
- ▶ Goals: User-friendliness, efficiency, safety



Travel info  
Navigation  
Calendar  
Infotainment  
...

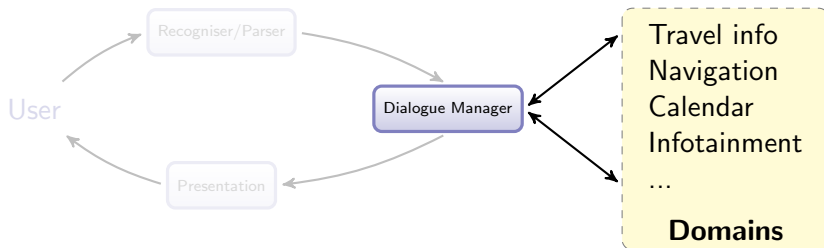
# Spoken-Language Dialogue Systems

- ▶ Goals: User-friendliness, efficiency, safety



# Spoken-Language Dialogue Systems

- ▶ Goals: User-friendliness, efficiency, safety



# Spoken-Language Dialogue Systems

- ▶ Goals: User-friendliness, efficiency, safety

## Command and Control

S: Choose your action!

U: New appointment.

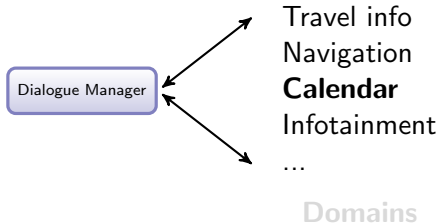
S: Ok. When does it start?

U: 9 AM.

S: When does it end?

U: 10 AM.

S: Sorry, this time is already taken.



# ... to Dialogue-Based Assistance

## Mobile scenario



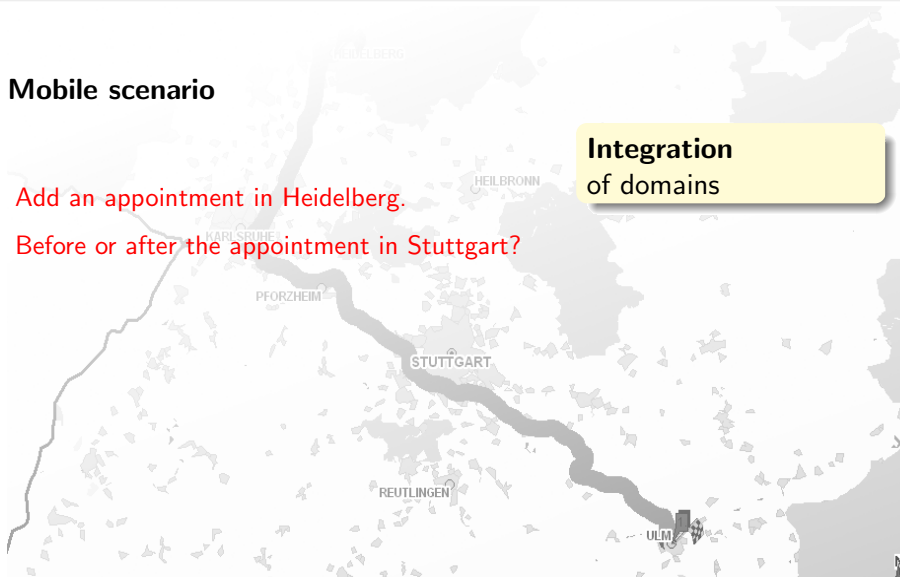
# ... to Dialogue-Based Assistance

## Mobile scenario

U: Add an appointment in Heidelberg.

S: Before or after the appointment in Stuttgart?

Integration  
of domains



# ... to Dialogue-Based Assistance

## Mobile scenario

U: Add an appointment in Heidelberg.

S: Before or after the appointment in Stuttgart?

U: How much time, if before?

S: 30 minutes.

Integration  
of domains

Handling **what-if**  
subdialogues

# ... to Dialogue-Based Assistance

## Mobile scenario

U: Add an appointment in Heidelberg.

S: Before or after the appointment in Stuttgart?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

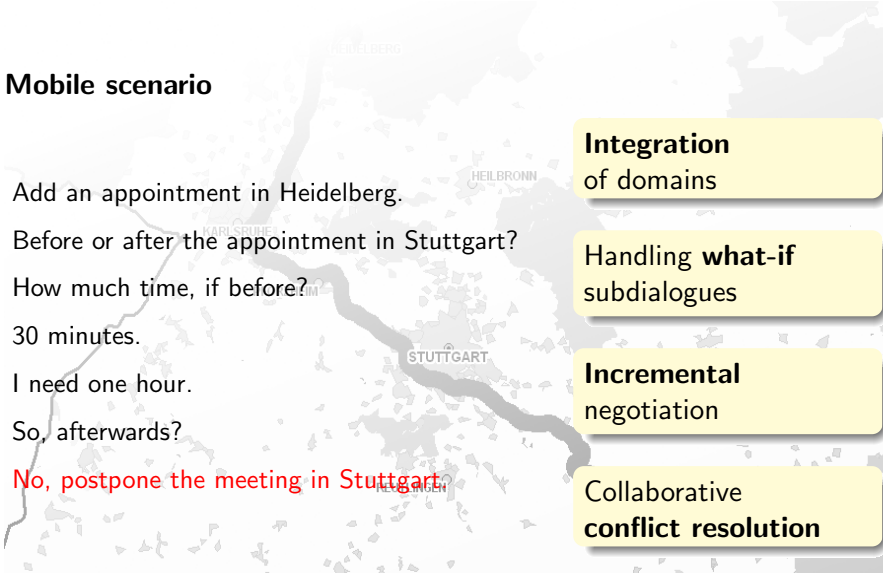
**Integration**  
of domains

Handling **what-if**  
subdialogues

**Incremental**  
negotiation

# ... to Dialogue-Based Assistance

## Mobile scenario

- 
- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in Stuttgart?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: **No, postpone the meeting in Stuttgart.**

**Integration**  
of domains

Handling **what-if**  
subdialogues

**Incremental**  
negotiation

Collaborative  
**conflict resolution**

# Problem Setting

- ▶ Current dialogue architectures handle some but not all of these interactions in a straightforward way.
  - ▶ cf. TRIPS, SmartKom, VoiceXML
- ▶ However, powerful domain-level reasoning approaches have been developed.
  - ▶ Theorem provers, planning systems, model generators
  - ▶ Interactivity is required

# Problem Setting

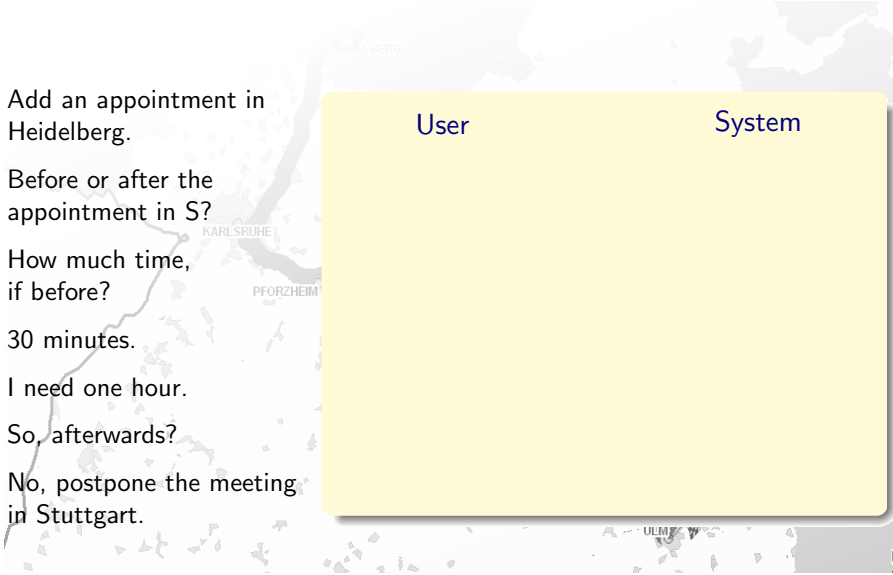
- ▶ Current dialogue architectures handle some but not all of these interactions in a straightforward way.
  - ▶ cf. TRIPS, SmartKom, VoiceXML
- ▶ However, powerful domain-level reasoning approaches have been developed.
  - ▶ Theorem provers, planning systems, model generators
  - ▶ Interactivity is required

**Goal: Bridging the gap between dialogue management and interactive reasoning.**

# Outline

- ▶ Introduction
  - ▶ Spoken-language dialogue systems
  - ▶ Motivation and problem setting
- ▶ Dialogue architecture for interactive reasoning
  - ▶ Dialogue management approach
  - ▶ Interactive reasoning engine
- ▶ Prototype systems
- ▶ Conclusions

# Dialogue Management Approach

- 
- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

**Requirements**

# Dialogue Management Approach

**U: Add an appointment in Heidelberg.**

**S:** Before or after the appointment in S?

**U:** How much time, if before?

**S:** 30 minutes.

**U:** I need one hour.

**S:** So, afterwards?

**U:** No, postpone the meeting in Stuttgart.

User

System

**Requirements**

# Dialogue Management Approach

U: **Add an appointment in Heidelberg.**

S: Before or after the appointment in S?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

U: No, postpone the meeting in Stuttgart.

User

System

Requirements

$app(a_2, HD)$

# Dialogue Management Approach

U: **Add an appointment in Heidelberg.**

S: Before or after the appointment in S?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

U: No, postpone the meeting in Stuttgart.

User

System

## Requirements

$\text{app}(a_1, S)$

$\text{app}(a_2, HD)$

# Dialogue Management Approach

U: **Add an appointment in Heidelberg.**

S: Before or after the appointment in S?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

U: No, postpone the meeting in Stuttgart.

User

**Requirements**

$\text{app}(a_1, S)$

$\text{app}(a_2, HD)$

System

**Inferences**

# Dialogue Management Approach

**U: Add an appointment in Heidelberg.**

**S:** Before or after the appointment in S?

**U:** How much time, if before?

**S:** 30 minutes.

**U:** I need one hour.

**S:** So, afterwards?

**U:** No, postpone the meeting in Stuttgart.

User

**Requirements**

$\text{app}(a_1, S)$

$\wedge$

$\text{app}(a_2, HD)$

System

**Inferences**

# Dialogue Management Approach

**U:** Add an appointment in Heidelberg.

**S:** Before or after the appointment in S?

**U:** How much time, if before?

**S:** 30 minutes.

**U:** I need one hour.

**S:** So, afterwards?

**U:** No, postpone the meeting in Stuttgart.

User

System

Requirements

Inferences

$\text{app}(a_1, S)$

$\wedge$

$\longrightarrow$   
axiom

$\text{app}(a_2, HD)$

# Dialogue Management Approach

**U:** Add an appointment in Heidelberg.

**S:** Before or after the appointment in S?

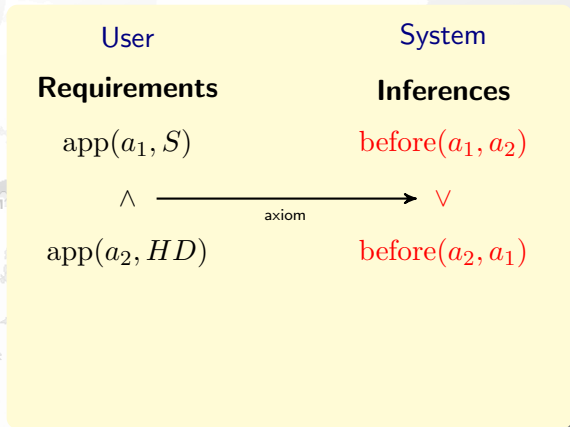
**U:** How much time, if before?

**S:** 30 minutes.

**U:** I need one hour.

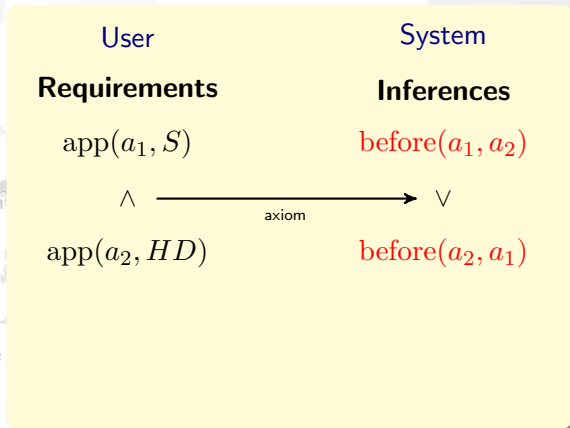
**S:** So, afterwards?

**U:** No, postpone the meeting in Stuttgart.



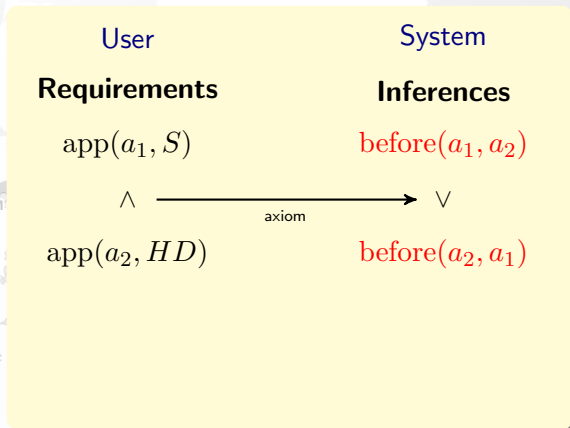
# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: **Before or after the appointment in S?**
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: **Before or after** the appointment in **S**?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: **Before or after the appointment in S?**
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

**Requirements**

**Inferences**

$\text{app}(a_1, S)$

$\text{before}(a_1, a_2)$

$\wedge \xrightarrow{\text{axiom}} \vee$

$\text{app}(a_2, HD)$

$\text{before}(a_2, a_1)$

► **Solution: consistent and unambiguous**

# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: **30 minutes.**
- U: **I need one hour.**
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

Requirements

Inferences

$\text{app}(a_1, S)$

$\text{before}(a_1, a_2)$

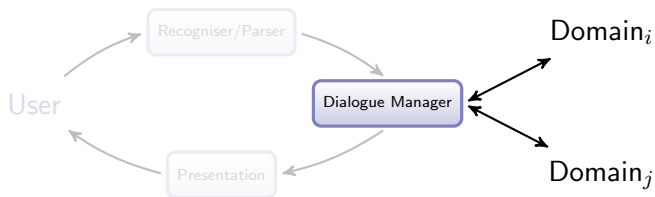
$\wedge \xrightarrow{\text{axiom}} \vee$

$\text{app}(a_2, HD)$

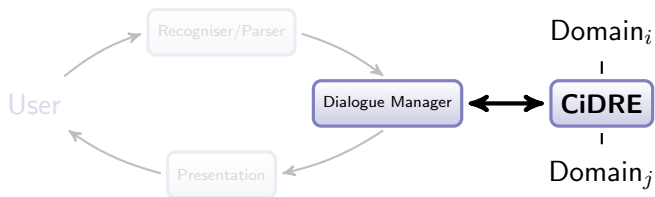
$\text{before}(a_2, a_1)$

- ▶ **Solution:** consistent and unambiguous
- ▶ **Conflict:** overconstrained requirements

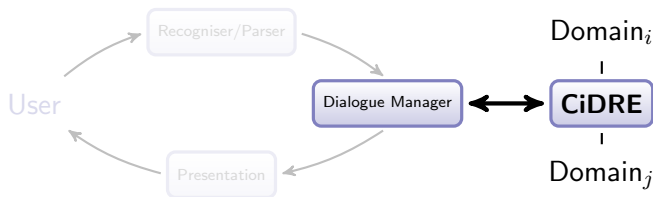
# Extended Architecture



# Extended Architecture



# Extended Architecture



- ▶ Common Interactive Domain-level Reasoning Engine (**CiDRE**)
- ▶ Responsible for providing a reasoning service to DM
- ▶ ... and a single interface to the domain back-end

# Prerequisites for Interactive Reasoning



# Prerequisites for Interactive Reasoning



- 1 **Interactive** and **incremental** processing of requirements and inferences

# Prerequisites for Interactive Reasoning



- 1 **Interactive** and **incremental** processing of requirements and inferences
- 2 **Transparency**: Dialogue needs **proof traces** (e.g. for resolving conflicting user requirements)

# Reasoning in CiDRE

Dialogue Manager

CiDRE

- ▶ **Finite model generation:**  $\Delta \models \Phi$   
 $\Phi$ : domain-specific system axioms + user requirements

# Reasoning in CiDRE

Dialogue Manager

CiDRE

- ▶ **Finite model generation:**  $\Delta \models \Phi$   
 $\Phi$ : domain-specific system axioms + user requirements
- ▶ Tableau method: Positive Unit Hyper Resolution (Bry 1996)

# Reasoning in CiDRE



- ▶ **Finite model generation:**  $\Delta \models \Phi$   
 $\Phi$ : domain-specific system axioms + user requirements
- ▶ Tableau method: Positive Unit Hyper Resolution (Bry 1996)
- ▶ Extensions in this work:
  - ▶ Reasoning procedure as an **iterative function**
  - ▶ **Interactive protocol** for Dialogue Manager
  - ▶ Generation and management of **proof traces**

# Outline

- ▶ Introduction
  - ▶ Spoken-language dialogue systems
  - ▶ Motivation and problem setting
- ▶ Dialogue architecture for interactive reasoning
  - ▶ Dialogue management approach
  - ▶ Interactive reasoning engine
- ▶ **Prototype systems**
  - ▶ **VoiceXML**
  - ▶ **Information states**
- ▶ Conclusions

# VoiceXML Prototype

- ▶ W3C standard
  - ▶ Dialogue description based on **forms**
  - ▶ Growing commercial relevance

# VoiceXML Prototype

- ▶ W3C standard
  - ▶ Dialogue description based on **forms**
  - ▶ Growing commercial relevance
- ▶ **Benefits** from CiDRE
  - ▶ Extend existing applications
  - ▶ Development: use inferences instead of “scripting”
  - ▶ Flexible user input; conflict handling

# VoiceXML Prototype

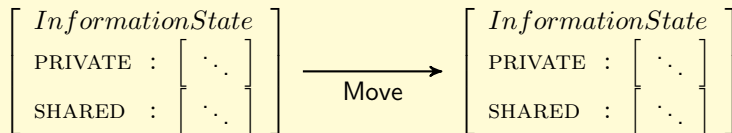
- ▶ W3C standard
  - ▶ Dialogue description based on **forms**
  - ▶ Growing commercial relevance
- ▶ **Benefits** from CiDRE
  - ▶ Extend existing applications
  - ▶ Development: use inferences instead of “scripting”
  - ▶ Flexible user input; conflict handling
- ▶ **Limitations**
  - ▶ Technical: Asynchronous communication with CiDRE
  - ▶ Conceptual restrictions of form representation

# Information States

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \end{array} \right]$$

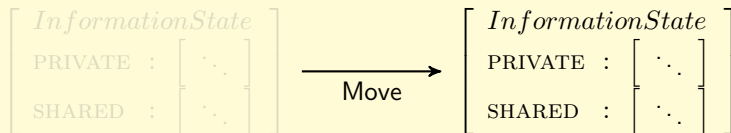
- ▶ **Information State:** formal and comprehensive representation of the information used in dialogue management

# Information States



- ▶ **Information State:** formal and comprehensive representation of the information used in dialogue management

# Information States



- ▶ **Information State:** formal and comprehensive representation of the information used in dialogue management
- ▶ **Dialogue Move:** **update** the information state (user and system)

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \end{array} \right]$$

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \end{array} \right]$$

- 1 Information state structure:

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \end{array} \right]$$

- 1 Information state structure:
- 2 Set of dialogue moves:

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \end{array} \right] \end{array} \right]$$

- 1 Information state structure:
- 2 Set of dialogue moves:
- 3 Selection and processing of moves.

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{l} \cdot \\ \cdot \\ \cdot \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{l} \text{REQ} : \\ \text{QUD} : \end{array} \right] \end{array} \right]$$

- 1 Information state structure:
  - ▶ Requirements, questions
- 2 Set of dialogue moves:
- 3 Selection and processing of moves.

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{l} \text{INF} : \\ \text{TASKS} : \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{l} \text{REQ} : \\ \text{QUD} : \end{array} \right] \end{array} \right]$$

- ① Information state structure:
  - ▶ Requirements, questions, inferences
- ② Set of dialogue moves:
- ③ Selection and processing of moves.

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{l} \text{INF} : \\ \text{TASKS} : \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{l} \text{REQ} : \\ \text{QUD} : \end{array} \right] \end{array} \right]$$

- ① Information state structure:
  - ▶ Requirements, questions, inferences
- ② Set of dialogue moves:
  - ▶ **ASSERT**, **RETRACT** requirements; **QUERY** for requests
- ③ Selection and processing of moves.

# Information States for Interactive Reasoning

$$\left[ \begin{array}{l} \textit{InformationState} \\ \text{PRIVATE} : \left[ \begin{array}{l} \text{INF} : \\ \text{TASKS} : \end{array} \right] \\ \text{SHARED} : \left[ \begin{array}{l} \text{REQ} : \\ \text{QUD} : \end{array} \right] \end{array} \right]$$

- ① Information state structure:
  - ▶ Requirements, questions, inferences
- ② Set of dialogue moves:
  - ▶ **ASSERT**, **RETRACT** requirements; **QUERY** for requests
- ③ Selection and processing of moves.
  - ▶ CiDRE interface and system strategy

# Processing Dialogue Moves

U: How much time, if before? S: 30 minutes.

## *InformationState*

PRIVATE :

INF :

TASKS :  $\langle r_0 \rangle$

SHARED :

REQ :  $\{ \text{app}(a_1),$   
 $\text{app}(a_2) \}$

QUD :

- ▶ U: ASSERT  $\text{before}(a_2, a_1)$
- ▶ U: QUERY  $x.\text{dur}(a_2, x)$
- ▶ S: ASSERT  $\text{dur}(a_2, 30m)$

# Processing Dialogue Moves

U: How much time, if before? S: 30 minutes.

## *InformationState*

PRIVATE :

INF :  $\emptyset$

TASKS :  $\langle r_0 \rangle$

SHARED :

REQ :  $\{ \text{app}(a_1),$   
 $\text{app}(a_2) \}$

QUD :  $\emptyset$

- ▶ U: ASSERT  $\text{before}(a_2, a_1)$
- ▶ U: QUERY  $x.\text{dur}(a_2, x)$
- ▶ S: ASSERT  $\text{dur}(a_2, 30m)$

# Processing Dialogue Moves

U: How much time, if before? S: 30 minutes.

## *InformationState*

PRIVATE :

INF :  $\emptyset$

TASKS :  $\langle r_0 \rangle$

SHARED :

REQ :  $\{ \text{app}(a_1),$   
 $\text{app}(a_2) \}$

QUD :  $\emptyset$

▶ U: ASSERT before( $a_2, a_1$ )

▶ U: QUERY  $x.\text{dur}(a_2, x)$

▶ S: ASSERT dur( $a_2, 30m$ )

# Processing Dialogue Moves

U: How much time, if before? S: 30 minutes.

## InformationState

PRIVATE :

INF :  $\emptyset$

TASKS :  $\langle r_0, r_1 \rangle$

SHARED :

REQ :  $\{ \text{app}(a_1),$   
 $\text{app}(a_2),$   
 $\text{before}(a_2, a_1) \}$

QUD :  $\emptyset$

- ▶ U: ASSERT **before**( $a_2, a_1$ )
  - ▶ Add requirement
  - ▶ Create reasoning task
- ▶ U: QUERY  $x.\text{dur}(a_2, x)$
- ▶ S: ASSERT  $\text{dur}(a_2, 30m)$

# Processing Dialogue Moves

U: How much time, if before? S: 30 minutes.

## InformationState

PRIVATE :

INF :  $\emptyset$

TASKS :  $\langle r_0, r_1, r_2 \rangle$

SHARED :

REQ :  $\{ \text{app}(a_1),$   
 $\text{app}(a_2),$   
 $\text{before}(a_2, a_1) \}$

QUD :  $\{ x.\text{dur}(a_2, x) \}$

- ▶ U: ASSERT  $\text{before}(a_2, a_1)$ 
  - ▶ Add requirement
  - ▶ Create reasoning task
- ▶ U: QUERY  $x.\text{dur}(a_2, x)$ 
  - ▶ Add QUD
  - ▶ Create reasoning task
  - ▶  $\forall x. \neg \text{dur}(a_2, x)$
  - ▶ Wait for CiDRE
- ▶ S: ASSERT  $\text{dur}(a_2, 30m)$

# Processing Dialogue Moves

U: How much time, if before? S: 30 minutes.

## InformationState

PRIVATE :

INF :  $\{\perp\}$

TASKS :  $\langle r_0, r_1 \rangle$

SHARED :

REQ :  $\{ \text{app}(a_1),$   
 $\text{app}(a_2),$   
 $\text{before}(a_2, a_1) \}$

QUD :  $\emptyset$

- ▶ U: ASSERT  $\text{before}(a_2, a_1)$ 
  - ▶ Add requirement
  - ▶ Create reasoning task
- ▶ U: QUERY  $x.\text{dur}(a_2, x)$ 
  - ▶ Add QUD
  - ▶ Create reasoning task
  - ▶  $\forall x. \neg \text{dur}(a_2, x)$
  - ▶ Wait for CiDRE
- ▶ S: ASSERT  $\text{dur}(a_2, 30m)$ 
  - ▶ Present answer
  - ▶ Drop reasoning task
  - ▶ Relax QUD

# Outline

- ▶ Introduction
  - ▶ Spoken-language dialogue systems
  - ▶ Motivation and problem setting
- ▶ Dialogue architecture for interactive reasoning
  - ▶ Dialogue management approach
  - ▶ Interactive reasoning engine
- ▶ Prototype systems
  - ▶ VoiceXML
  - ▶ Information states
- ▶ **Conclusions**
  - ▶ **Contributions**
  - ▶ **Future Directions**

# Contributions

## Theoretical:

- ▶ Dialogue management approach based on interactive reasoning
- ▶ Modelling of application domains

## Practical:

- ▶ Dialogue systems architecture
- ▶ Interactive reasoning engine (CiDRE)

## Experimental:

- ▶ System prototypes: VoiceXML, information states
- ▶ Domains: TRAINS logistics, calendar, restaurants and others

# Future Directions

Dialogue management:

- ▶ User preferences, profiles, adaptivity

Reasoning and representation:

- ▶ Dealing with soft constraints

Systems:

- ▶ Full system evaluation, with non-experts

# Publications

- Bühler, Minker (2007). HMM-Based Semantic Analysis for the ESST and MEDIA Tasks. In *IE 07*.
- Bühler, Minker (2006). Stochastic Spoken Natural Language Parsing in the Framework of the French MEDIA Evaluation Campaign. In *LREC*.
- Bühler, Hamerich (2005). Towards VoiceXML Compilation for Portable Embedded Applications in Ubiquitous Environments. In *EUROSPEECH*.
- Bühler, Hernández, Minker (2005). Connecting Dialogue Systems to the Internet. In *IE05*.
- Bühler, Minker (2005). *A Reasoning Component for Information-Seeking and Planning Dialogues*, vol. 28 of *Text, Speech and Language Technology*. Springer.
- Bühler, Minker (2005). Mobile Multimodality - Design and Development of the SmartKom Companion. *International Journal of Speech Technology*, 8(2):193–202.
- Bühler, Minker (2005). *The SmartKom Mobile Car Prototype System for Flexible Human-Machine Communication*, vol. 28 of *Text, Speech and Language Technology*. Springer.
- Bühler, Minker, Elciyanti (2005). Using Language Modelling to Integrate Speech Recognition with a Flat Semantic Analysis. In *SIGdial*.
- Bühler, Riegler (2005). Integrating Dialogue Management and Domain Reasoning. In *SPECOM*.
- Bühler (2004). Enhancing Existing Form-Based Dialogue Managers with Reasoning Capabilities. In *ICSLP*.
- Bühler, Hamerich (2004). Towards Embedding VoiceXML Applications through Compilation. In *Berliner XML-Tage*.
- Bühler, Vignier, Heisterkamp, Minker (2003). Safety and Operating Issues for Mobile Human-Machine Interfaces. In *IUI*.
- Bühler, Minker (2002). An Architecture for a Logic-Based Human-Machine Dialogue. In *ITRW*.
- Bühler, Minker, Häussler, Krüger (2002). Flexible Multimodal Human-Machine Interaction in Mobile Environments. In *ICSLP*.

# Publications

- Bühler, Minker (2007). HMM-Based Semantic Analysis for the ESST and MEDIA Tasks. In *IE 07*.
- Bühler, Minker (2006). Stochastic Spoken Natural Language Parsing in the Framework of the French MEDIA Evaluation Campaign. In *LREC*.
- Bühler, Hamerich (2005). Towards VoiceXML Compilation for Portable Embedded Applications in Ubiquitous Environments. In *EUROSPEECH*.
- Bühler, Hernández, Minker (2005). Connecting Dialogue Systems to the Internet. In *IE05*.
- Bühler, Minker (2005). *A Reasoning Component for Information-Seeking and Planning Dialogues*, vol. 28 of *Text, Speech and Language Technology*. Springer.
- Bühler, Minker (2005). Mobile Multimodality - Design and Development of the SmartKom Companion. *International Journal of Speech Technology*, 8(2):193–202.
- Bühler, Minker (2005). *The SmartKom Mobile Car Prototype System for Flexible Human-Machine Communication*, vol. 28 of *Text, Speech and Language Technology*. Springer.
- Bühler, Minker, Elciyanti (2005). Using Language Modelling to Integrate Speech Recognition with a Flat Semantic Analysis. In *SIGdial*.
- Bühler, Riegler (2005). Integrating Dialogue Management and Domain Reasoning. In *SPECOM*.
- Bühler (2004). Enhancing Existing Form-Based Dialogue Managers with Reasoning Capabilities. In *ICSLP*.
- Bühler, Hamerich (2004). Towards Embedding VoiceXML Applications through Compilation. In *Berliner XML-Tage*.
- Bühler, Vignier, Heisterkamp, Minker (2003). Safety and Operating Issues for Mobile Human-Machine Interfaces. In *IUI*.
- Bühler, Minker (2002). An Architecture for a Logic-Based Human-Machine Dialogue. In *ITRW*.
- Bühler, Minker, Häussler, Krüger (2002). Flexible Multimodal Human-Machine Interaction in Mobile Environments. In *ICSLP*.

# Publications

- Bühler, Minker (2007). HMM-Based Semantic Analysis for the ESST and MEDIA Tasks. In *IE 07*.
  - Bühler, Minker (2006). Stochastic Spoken Natural Language Parsing in the Framework of the French MEDIA Evaluation Campaign. In *LREC*.
  - Bühler, Hamerich (2005). Towards VoiceXML Compilation for Portable Embedded Applications in Ubiquitous Environments. In *EUROSPEECH*.
  - Bühler, Hernández, Minker (2005). Connecting Dialogue Systems to the Internet. In *IE05*.
  - Bühler, Minker (2005). *A Reasoning Component for Information-Seeking and Planning Dialogues*, vol. 28 of *Text, Speech and Language Technology*. Springer.
  - Bühler, Minker (2005). Mobile Multimodality - Design and Development of the SmartKom Companion. *International Journal of Speech Technology*, 8(2):193–202.
  - Bühler, Minker (2005). *The SmartKom Mobile Car Prototype System for Flexible Human-Machine Communication*, vol. 28 of *Text, Speech and Language Technology*. Springer.
  - Bühler, Minker, Elciyanti (2005). Using Language Modelling to Integrate Speech Recognition with a Flat Semantic Analysis. In *SIGdial*.
  - Bühler, Riegler (2005). Integrating Dialogue Management and Domain Reasoning. In *SPECOM*.
  - Bühler (2004). Enhancing Existing Form-Based Dialogue Managers with Reasoning Capabilities. In *ICSLP*.
  - Bühler, Hamerich (2004). Towards Embedding VoiceXML Applications through Compilation. In *Berliner XML-Tage*.
  - Bühler, Vignier, Heisterkamp, Minker (2003). Safety and Operating Issues for Mobile Human-Machine Interfaces. In *IUI*.
  - Bühler, Minker (2002). An Architecture for a Logic-Based Human-Machine Dialogue. In *ITRW*.
  - Bühler, Minker, Häussler, Krüger (2002). Flexible Multimodal Human-Machine Interaction in Mobile Environments. In *ICSLP*.
- + co-authored: 3 workshop, 9 conference, 1 journal, 3 book chapters, 1 book edited, 1 tech report

Thank you for your attention!

# Modelling an Application Domain

At run time: “current problem” + **domain theory**:

- ▶ **Structural description**: concepts, entities
- ▶ **Dynamic description**: temporal aspects

## APPOINTMENTS domain theory

$$\text{appointment}(a_1) \wedge \text{appointment}(a_2) \rightarrow \\ a_1 = a_2 \vee \text{before}(a_1, a_2) \vee \text{before}(a_2, a_1).$$

These concepts form a reusable library of domain theories.

# VoiceXML Limitations

Form

Requirements

Inferences

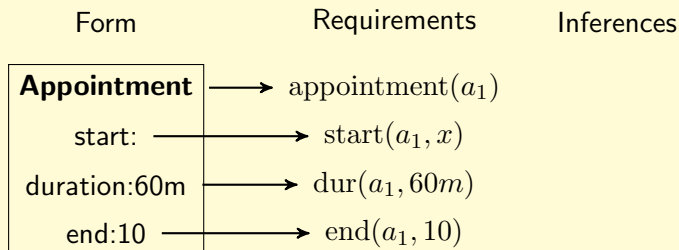
**Appointment**

start:

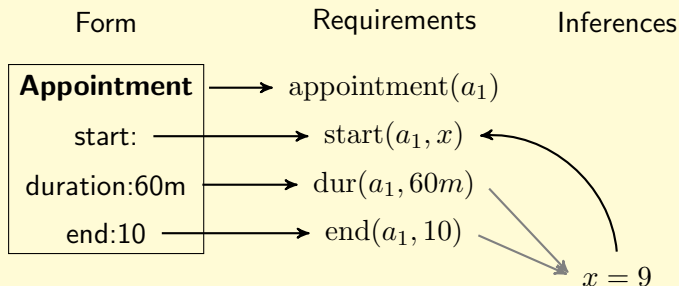
duration:60m

end:10

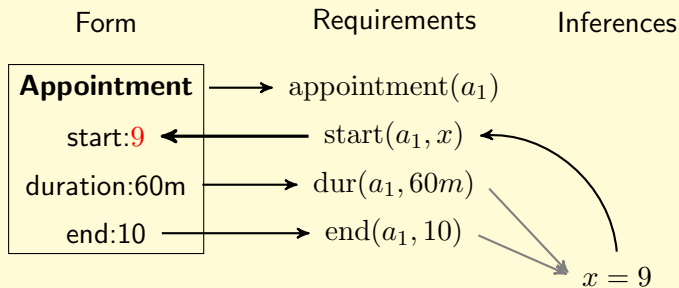
# VoiceXML Limitations



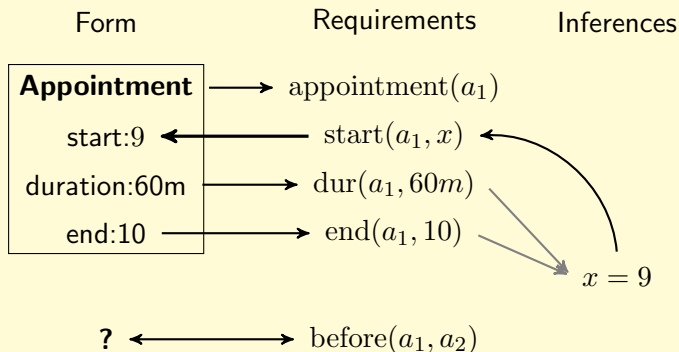
# VoiceXML Limitations



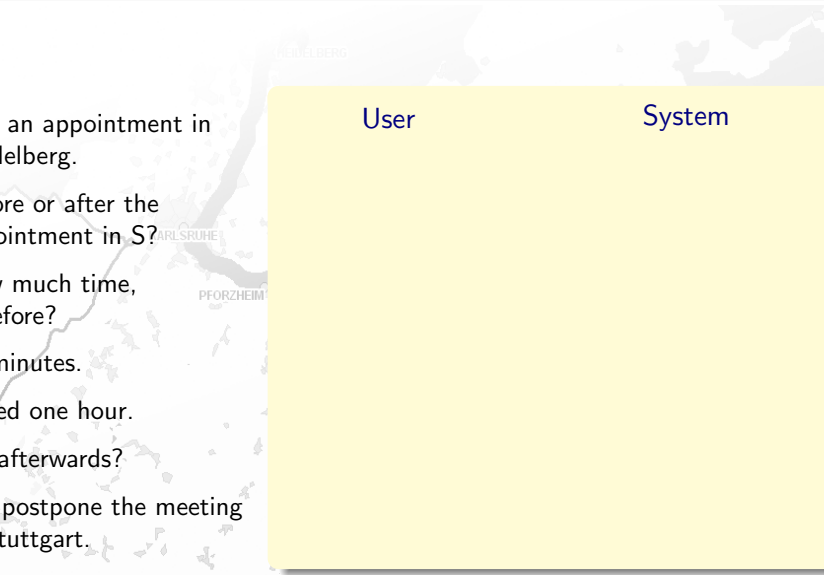
# VoiceXML Limitations



# VoiceXML Limitations



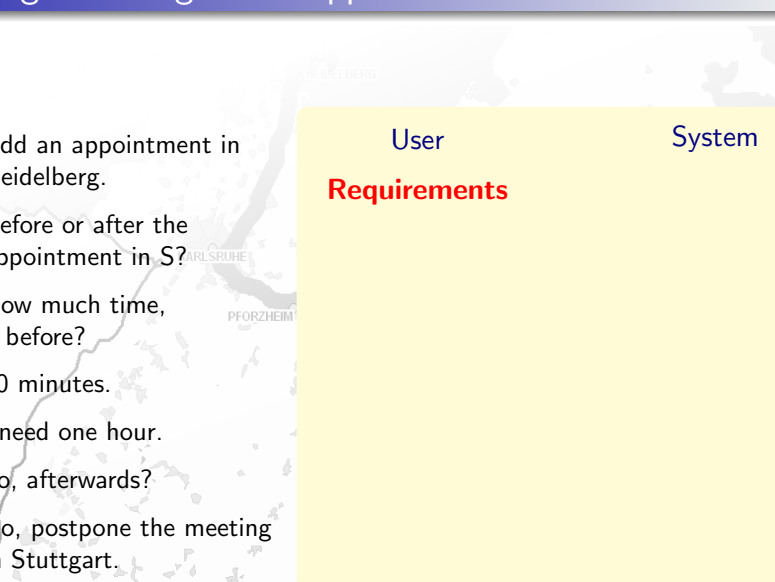
# Dialogue Management Approach

- 
- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

# Dialogue Management Approach

- 
- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

**Requirements**

# Dialogue Management Approach

**U: Add an appointment in Heidelberg.**

**S:** Before or after the appointment in S?

**U:** How much time, if before?

**S:** 30 minutes.

**U:** I need one hour.

**S:** So, afterwards?

**U:** No, postpone the meeting in Stuttgart.

User

System

**Requirements**

# Dialogue Management Approach

U: **Add an appointment in Heidelberg.**

S: Before or after the appointment in S?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

U: No, postpone the meeting in Stuttgart.

User

System

Requirements

$app(a_2, HD)$

# Dialogue Management Approach

U: **Add an appointment in Heidelberg.**

S: Before or after the appointment in S?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

U: No, postpone the meeting in Stuttgart.

User

System

## Requirements

$\text{app}(a_1, S)$

$\text{app}(a_2, HD)$

# Dialogue Management Approach

U: **Add an appointment in Heidelberg.**

S: Before or after the appointment in S?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

U: No, postpone the meeting in Stuttgart.

User

**Requirements**

$\text{app}(a_1, S)$

$\text{app}(a_2, HD)$

System

**Inferences**

# Dialogue Management Approach

**U: Add an appointment in Heidelberg.**

**S:** Before or after the appointment in *S*?

**U:** How much time, if before?

**S:** 30 minutes.

**U:** I need one hour.

**S:** So, afterwards?

**U:** No, postpone the meeting in Stuttgart.

User

Requirements

$\text{app}(a_1, S)$

$\text{app}(a_2, HD)$

System

Inferences



# Dialogue Management Approach

U: **Add an appointment in Heidelberg.**

S: Before or after the appointment in S?

U: How much time, if before?

S: 30 minutes.

U: I need one hour.

S: So, afterwards?

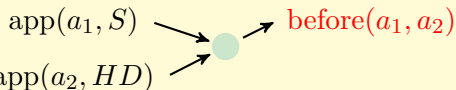
U: No, postpone the meeting in Stuttgart.

User

System

Requirements

Inferences



# Dialogue Management Approach

**U: Add an appointment in Heidelberg.**

**S:** Before or after the appointment in *S*?

**U:** How much time, if before?

**S:** 30 minutes.

**U:** I need one hour.

**S:** So, afterwards?

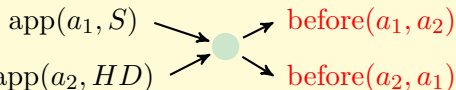
**U:** No, postpone the meeting in Stuttgart.

User

System

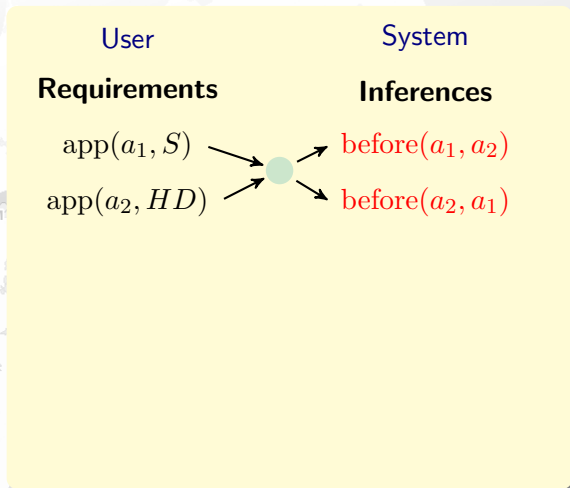
Requirements

Inferences



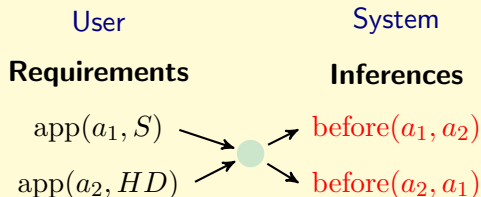
# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: **Before or after the appointment in S?**
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: **Before or after the appointment in S?**
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



# Dialogue Management Approach

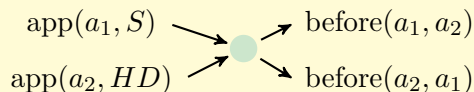
- U: Add an appointment in Heidelberg.
- S: **Before or after the appointment in S?**
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

Requirements

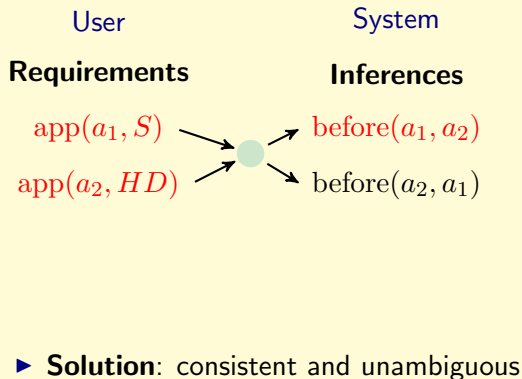
Inferences



► **Solution:** consistent and unambiguous

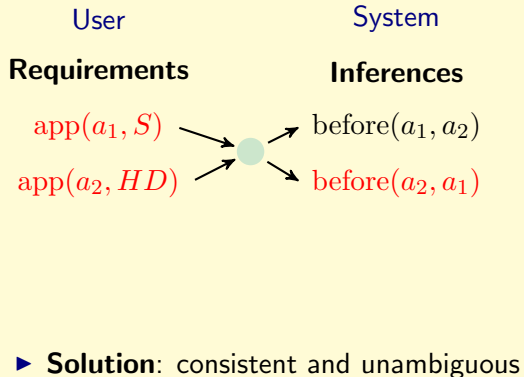
# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: **Before or after the appointment in S?**
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: **Before or after the appointment in S?**
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



# Dialogue Management Approach

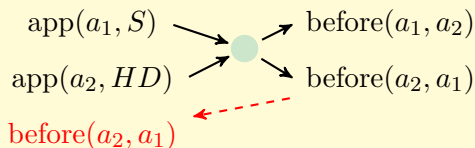
- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: **How much time, if before?**
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.

User

System

Requirements

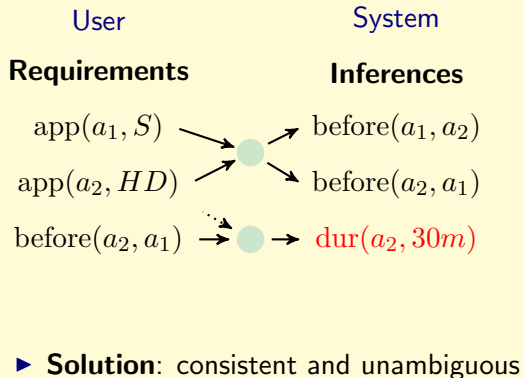
Inferences



► **Solution:** consistent and unambiguous

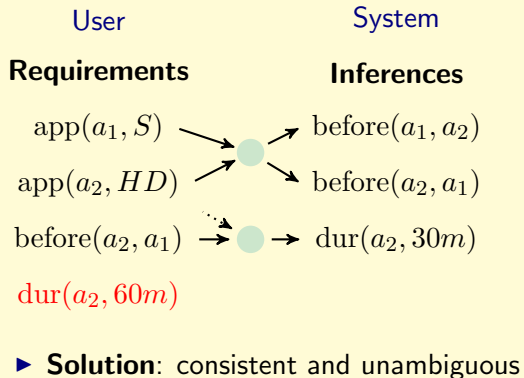
# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: **30 minutes.**
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



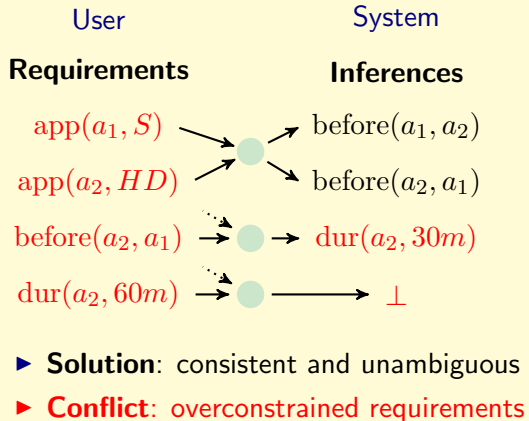
# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: 30 minutes.
- U: I need **one hour**.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



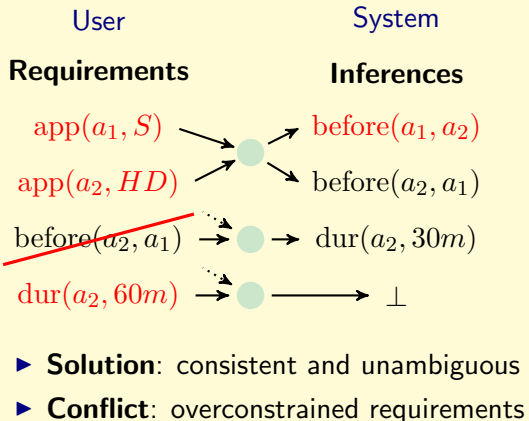
# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: So, afterwards?
- U: No, postpone the meeting in Stuttgart.



# Dialogue Management Approach

- U: Add an appointment in Heidelberg.
- S: Before or after the appointment in S?
- U: How much time, if before?
- S: 30 minutes.
- U: I need one hour.
- S: **So, afterwards?**
- U: No, postpone the meeting in Stuttgart.



$$A(n, m) := \{\top \rightarrow a[i, 1] \vee \dots \vee a[i, m] \mid i = 1, \dots, n\}$$

$$B(n, m) := C(n, m, m^n) \cup A(n, m)$$

$$C(n, m, k) := \{M[j] \rightarrow \perp \mid j = 1, \dots, k - 1\}$$

$$D(n, m, k) := E(n, m, k) \cup A(n, m)$$

$$E(n, m, k) := \{a[i, j] \rightarrow a[i + 1, j] \mid i = 1, \dots, n - 1, j = 1, \dots, k\}$$

$$F(n, m, k) := C(n, m, m^n) \cup E(n, m, k) \cup A(n, m)$$

$m$	3	4	5	6	7	8	9	10
$A(5, m)$	0.15	1.44	13.74	85.05	413.62	1 445.91	4 782.78	13 317.30
$B(5, m)$	0.10	1.29	14.99	102.43	451.75	1 799.27	5 737.51	15 398.20
$D(5, m, 1)$	0.07	0.63	5.19	42.13	299.14	928.12	3 165.38	8 819.52
$D(5, m, \lfloor \frac{m}{2} \rfloor)$	0.07	0.29	1.84	4.38	28.98	57.45	315.45	542.96
$D(5, m, m - 1)$	0.05	0.28	0.63	1.59	3.51	7.09	13.14	22.88
$F(5, m, 1)$	0.12	1.40	16.60	100.38	462.95	1 688.92	5 704.90	15 242.60
$F(5, m, \lfloor \frac{m}{2} \rfloor)$	0.12	0.97	12.69	60.84	303.68	989.93	3 585.84	8 608.44
$F(5, m, m - 1)$	0.08	0.59	6.06	32.30	134.27	482.85	1 498.90	4 202.49

**Table:** Summary of CPU times in seconds for MM-SATCHMO (Bry 2000).

$m$	3	4	5	6	7	8	9	10
$A(5, m)$	0.04	0.18	0.64	1.49	3.34	6.73	12.87	22.46
$B(5, m)$	0.18	1.12	7.22	31.92	120.25	376.14	1 064.83	2 687.09
$D(5, m, 1)$	0.01	0.08	0.27	0.80	1.99	4.29	8.30	15.24
$D(5, m, \lfloor \frac{m}{2} \rfloor)$	0.01	0.02	0.12	0.14	0.46	0.54	1.47	1.65
$D(5, m, m - 1)$	0.00	0.00	0.01	0.01	0.02	0.02	0.02	0.02
$F(5, m, 1)$	0.09	0.65	3.81	18.27	71.32	234.33	692.21	1 816.32
$F(5, m, \lfloor \frac{m}{2} \rfloor)$	0.07	0.41	1.36	3.79	8.50	17.42	33.71	61.83
$F(5, m, m - 1)$	0.09	0.47	2.32	6.00	22.79	45.04	148.18	256.96

Table: Summary of CPU times in seconds for CiDRE.

$$L(n) := \{list(l), length(l) = n\}$$

$n$	0	1	2	3	4	5	10	15	20
CPU time [s]	1.61	1.68	1.70	1.70	1.75	1.78	2.07	2.60	3.37

Table: CPU processing times for  $L(n)$ .

$$FC(n) := \{value_f(i, i) \mid i = 0, \dots, n\}$$

$n$	0	1	2	3	4	5	10	15	20
CPU time [s]	1.23	1.54	2.00	2.47	3.26	4.12	17.21	71.35	249.78

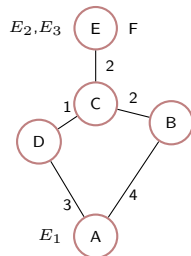
Table: CPU processing times for  $FC(n)$ .

n	1	2	7	8	62	63	300
CPU time [s]	52.85	56.31	165.01	167.89	4 242.39	4 248.99	287 047.11
Size	7 567	7 567	7 682	7 567	7 823	7 567	7 963
d-hours	1	2	2	3	3	4	4
Depth	1	1-9		1-22		1-34	
<i>n<sub>inc</sub></i>	0	6	22	35	996	998	8 466

Table: Measurements for enumerating TRAINS models.

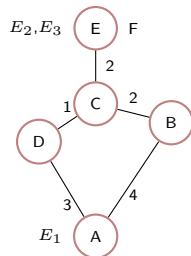
# Sample Domain: TRAINS

- ▶ Toy logistics domain (but not trivial)



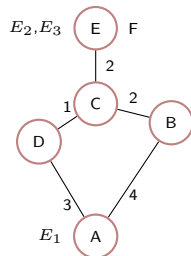
# Sample Domain: TRAINS

- ▶ Toy logistics domain (but not trivial)
- ▶ **Entities:** cities, engines, boxcars, commodities



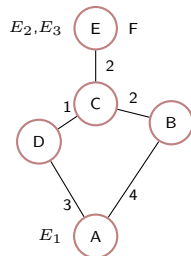
# Sample Domain: TRAINS

- ▶ Toy logistics domain (but not trivial)
- ▶ **Entities:** cities, engines, boxcars, commodities
- ▶ **Relations:** *city/1, at/3*



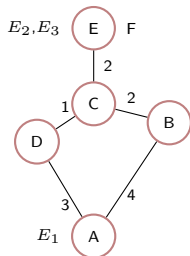
# Sample Domain: TRAINS

- ▶ Toy logistics domain (but not trivial)
- ▶ **Entities:** cities, engines, boxcars, commodities
- ▶ **Relations:** *city/1, at/3*
- ▶ **Ontology:** mobile/immobile, engine/container/commodity



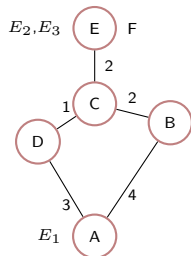
# Sample Domain: TRAINS

- ▶ Toy logistics domain (but not trivial)
- ▶ **Entities:** cities, engines, boxcars, commodities
- ▶ **Relations:** *city/1, at/3*
- ▶ **Ontology:** mobile/immobile, engine/container/commodity
- ▶ **Rules**
  - ▶ In order to be moved, a commodity has to be loaded in a boxcar, which in turn has to be attached to an engine
  - ▶ moving, loading/unloading takes time and occupies resources



# Sample Domain: TRAINS

- ▶ Toy logistics domain (but not trivial)
- ▶ **Entities:** cities, engines, boxcars, commodities
- ▶ **Relations:** *city/1, at/3*
- ▶ **Ontology:** mobile/immobile, engine/container/commodity
- ▶ **Rules**
  - ▶ In order to be moved, a commodity has to be loaded in a boxcar, which in turn has to be attached to an engine
  - ▶ moving, loading/unloading takes time and occupies resources



# Formalisation using First-Order Logic

– Ontology rules:

$engine(E_1). city(A). \dots$

$mobile(x) \rightarrow engine(x) \vee transporter(x) \vee product(x).$

$engine(x) \wedge transporter(x) \rightarrow.$

...

# Formalisation using First-Order Logic

– Ontology rules:

$engine(E_1). city(A). \dots$

$mobile(x) \rightarrow engine(x) \vee transporter(x) \vee product(x).$

$engine(x) \wedge transporter(x) \rightarrow.$

...

# Formalisation using First-Order Logic

– Ontology rules:

*engine*( $E_1$ ). *city*( $A$ ). ...

*mobile*( $x$ )  $\rightarrow$  *engine*( $x$ )  $\vee$  *transporter*( $x$ )  $\vee$  *product*( $x$ ).

*engine*( $x$ )  $\wedge$  *transporter*( $x$ )  $\rightarrow$ .

...

# Formalisation using First-Order Logic

– Ontology rules:

$engine(E_1). city(A). \dots$

$mobile(x) \rightarrow engine(x) \vee transporter(x) \vee product(x).$

$engine(x) \wedge transporter(x) \rightarrow.$

...

– Rules involving temporal aspects:

$at(x, t_1, l_1) \wedge at(x, t_2, l_2) \wedge before(t_1, t_2)$   
 $\rightarrow move(x, t_1, t_2) \vee nomove(x, t_1, t_2).$

# Formalisation using First-Order Logic

– Ontology rules:

$engine(E_1). city(A). \dots$

$mobile(x) \rightarrow engine(x) \vee transporter(x) \vee product(x).$

$engine(x) \wedge transporter(x) \rightarrow.$

...

– Rules involving temporal aspects:

$at(x, t_1, l_1) \wedge at(x, t_2, l_2) \wedge before(t_1, t_2)$

$\rightarrow move(x, t_1, t_2) \vee nomove(x, t_1, t_2).$

# Formalisation using First-Order Logic

– Ontology rules:

$engine(E_1). city(A). \dots$

$mobile(x) \rightarrow engine(x) \vee transporter(x) \vee product(x).$

$engine(x) \wedge transporter(x) \rightarrow.$

...

– Rules involving temporal aspects:

$at(x, t_1, l_1) \wedge at(x, t_2, l_2) \wedge before(t_1, t_2)$

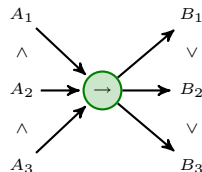
$\rightarrow move(x, t_1, t_2) \vee nomove(x, t_1, t_2).$

$nomove(x, t_1, t_2) \wedge at(x, t_1, l_1) \wedge at(x, t_2, l_2) \rightarrow l_1 = l_2.$

...

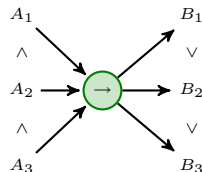
# Formalisation using First-Order Logic

- ▶ Basic syntax: First-order logic **clauses**
  - ▶ Pre and post conditions
  - ▶ Implicit  $\forall$  quantification
  - ▶  $\exists$  quantification through Skolem functions
  - ▶ Range restriction



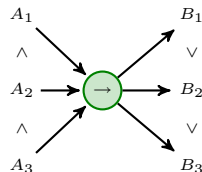
# Formalisation using First-Order Logic

- ▶ Basic syntax: First-order logic **clauses**
  - ▶ Pre and post conditions
  - ▶ Implicit  $\forall$  quantification
  - ▶  $\exists$  quantification through Skolem functions
  - ▶ Range restriction
- ▶ Construct axiomatization: a **domain theory** is a collection of clauses.



# Formalisation using First-Order Logic

- ▶ Basic syntax: First-order logic **clauses**
  - ▶ Pre and post conditions
  - ▶ Implicit  $\forall$  quantification
  - ▶  $\exists$  quantification through Skolem functions
  - ▶ Range restriction
- ▶ Construct axiomatization: a **domain theory** is a collection of clauses.
- ▶ Concepts from logics: model, interpretation, satisfiability



# Generalization using Fluents

- ▶ Temporally changing properties of an entity: **fluent**
- ▶ A fluent has exactly one value at a time
- ▶ **fluent change**

Old:  $at(x, t, l)$

New:  $fluentValue(position(x), t, l)$

Old:  $move(x, t_1, t_2)$

New:  $fluentChange(position(x), t_1, t_2)$

# Generalization using Fluents

- ▶ Temporally changing properties of an entity: **fluent**
- ▶ A fluent has exactly one value at a time
- ▶ **fluent change**

Old: *at*( $x, t, l$ )

New: *fluentValue*(*position*( $x$ ),  $t, l$ )

Old: *move*( $x, t_1, t_2$ )

New: *fluentChange*(*position*( $x$ ),  $t_1, t_2$ )

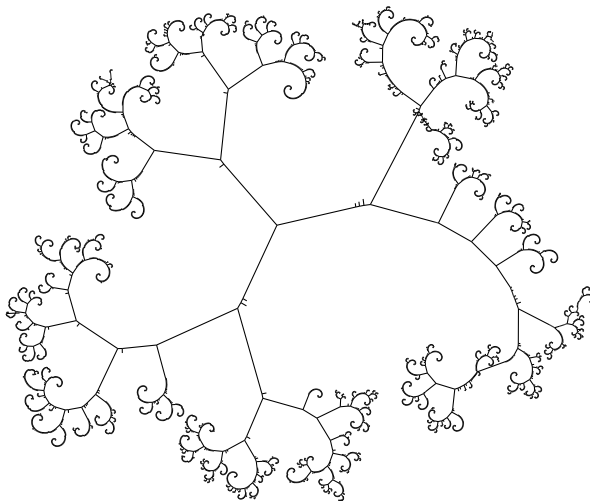
# Generalization using Fluents

- ▶ Temporally changing properties of an entity: **fluent**
- ▶ A fluent has exactly one value at a time
- ▶ **fluent change**

Old:  $at(x, t, l)$       New:  $fluentValue(position(x), t, l)$   
Old:  $move(x, t_1, t_2)$       New:  $fluentChange(position(x), t_1, t_2)$

- ▶ Generalize to other properties: loading, unloading of cargo, processing commodities.

# “The Big Picture”



- ▶ Search space of a reasoning process

