Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit.

Staffan Larsson & David Traum

(06/2000)

Überblick

- Information State Theory
- Formale Repräsentation
- Dialogue Moves
- Update Rules
- Update Strategy
- TRINDI Kit
- GoDIS
- Andere TRNDI Kit - Systeme

Motivation

- Modell zum Erstellen von DM-Komponenten
- Basis: Information State
- Extrahieren relevanter Information aus einem Dialog
- Wie wird Information aktualisiert ?
- Wie werden die Aktualisierungsprozesse kontrolliert ?

Information State Approach Theory

- Informational components
- Formal representations
- Dialogue moves
- Update rules
- Update strategy

Information State / Dialogue State

Structural Dialogue Modelling:

- “legal” dialogue: Verhalten nach vorgeschriebenen Regeln
- Jeder Zustand/State hat eine vorgeschriebene Anzahl zulässiger Übergänge zu anderen Zuständen

Plan-based Dialogue Modelling:

- Hohe Anzahl prozeduraler Verarbeitungsschritte
- Fehlen semantischer Grundlagen

Information State Dialogue Modelling:

- nutzt Vorteile der Structural- und Plan-based Ansätze
- andere Sichtweise auf Beliefs, Intentions, Plans

Informational Components

- Welche Komponenten sollen genutzt werden ?
- **Vielfältige Möglichkeiten**:
  - interne / externe Aspekte
  - mental state / structural view
  - statische / dynamische Aspekte
  - private / shared information
- **Auswahl abhängig von**
  - Domäne
  - Dialogform
    (question-answering vs. knowledge acquisition)

Private / Shared Information

Jonathan Ginzburg:

- **private** information:
  - Beliefs (BEL)
  - Actions to be performed (AGENDA)
- **shared** information
  - Propositions assumed to be shared beliefs (BEL)
  - Questions under discussion (QUD)
  - Latest dialogue move performed (LM)

Formal Representations

- Wie kann man die „internal states“ der Dialogteilnehmer modellieren ?
- **Viele Varianten**:
  - einfache abstrakte Datentypen / komplexe Systeme
  - statistische / logische Verfahren
  - set of beliefs / list of beliefs
  - Listen: FIFO Queue / LIFO Stack
- Auswahl abhängig von der Dialogstruktur

Formal Representations - Example

Information state dargestellt als **Record**

- BEL: Beliefs
- AGENDA: Actions to perform
- QUD: Questions under discussion
- LM: Latest dialogue move

$$\begin{bmatrix} \text{PRIVATE} : & \begin{bmatrix} \text{BEL} : \textsc{Set}(\textsc{Prop}) \\ \text{AGENDA} : \textsc{Stack}(\textsc{Action}) \end{bmatrix} \\ \text{SHARED} : & \begin{bmatrix} \text{BEL} : \textsc{Set}(\textsc{Prop}) \\ \text{QUD} : \textsc{Stack}(\textsc{Question}) \\ \text{LM} : \textsc{Move} \end{bmatrix} \end{bmatrix}$$

Dialogue Moves

- Abstraktion für die große Anzahl möglicher **Äußerungen**
- Abstraktionsebene für **Content**-Generierung
- Mindestanzahl an Dialogue moves notwendig, um einen Dialog **sinnvoll** gestalten zu können
- **Problem**: Adäquate Auswahl der Dialogue moves:
  - komplexe, multifunktionale Moves
  - einfache Moves
  - Moves im Sinne von Searles Sprechakten

Update Rules

- Formalisierung der **Änderungen** im Information State während des Dialogs
- U-Rule Zusammensetzung:
  - **Bedingungen**: an den Information State bzw. Information State muß U-Rule erzwingen
  - **Effekte**: Änderungen im Information State nach Anwendung der U-Rule
- U-Rules können auch als Bündel aus Änderungen im Information State betrachtet werden

Update Rules - Example

- Dialogue Moves: **ASK**, **ANSWER**
- U-Rule: **integrateSysAsk**
  Fügt eine Frage zu QUD, falls ASK ausgeführt wurde

$$\begin{array}{l}\textsc{u-rule: } \mathbf{integrateSysAsk} \\ \textsc{pre: } \begin{cases} \mathrm{val}(\textsc{shared.lm}, \mathrm{ask}(\mathrm{sys}, Q)) \\ \mathrm{fst}(\textsc{private.agenda}, \text{raise-question}(Q)) \end{cases} \\ \textsc{eff: } \begin{cases} \mathrm{push}(\textsc{shared.qud}, Q) \\ \mathrm{pop}(\textsc{private.agenda}) \end{cases} \end{array}$$

- U-Rule: **integrateUserAnswer**
  Entfernt die Frage von QUD, falls ANSWER ausgeführt wurde

$$\begin{array}{l}\textsc{u-rule: } \mathbf{integrateUserAnswer} \\ \textsc{pre: } \begin{cases} \mathrm{val}(\textsc{shared.lm}, \mathrm{answer}(\mathrm{usr}, A)), \\ \mathrm{in}(\textsc{shared.qud}, Q) \\ \text{answer-to}(A, Q) \\ \mathrm{apply}(Q, A, P) \end{cases} \\ \textsc{eff: } \begin{cases} \mathrm{pop}(\textsc{shared.qud}, Q), \\ \mathrm{add}(\textsc{shared.bel}, P) \end{cases} \end{array}$$

Update Strategy - Types

- Strategie, wie und wann die U-Rules angewendet werden
- Beeinflusst jedoch auch den Aufbau der U-Rules
- Einige U-Strategy Typen:
    1. *Take the first rule that applies (iteratively until no rules apply)*
    2. *Apply each rule (if applicable) in sequence*
    3. *Apply rules according to class*
    4. *Choose among applicable rules using probabilistic information*
    5. *Present choices to user to decide*

Update Strategy - Example

(4) [PRIVATE = [AGENDA = ⟨ RAISE(X^(TO=X) ⟩]]

Sys: Where do you want to go?

(triggers **integrateSysAsk**)

$$\begin{bmatrix} \text{PRIVATE} = [\ \text{AGENDA} = \langle\rangle\ ] \\ \text{SHARED} = \begin{bmatrix} \text{QUD} = \langle\ \text{X}\hat{}(\text{TO}=\text{X})\ \rangle \\ \text{LM} = \text{ASK}(\ \text{X}\hat{}(\text{TO}=\text{X})\ ) \end{bmatrix} \end{bmatrix}$$

Usr: Malvern

(triggers **integrateUsrAnswer**)

$$\begin{bmatrix} \text{SHARED} = \begin{bmatrix} \text{BEL} = \{\ (\text{TO}=\text{MALVERN})\ \} \\ \text{QUD} = \langle\rangle \\ \text{LM} = \text{ANSWER}(\text{MALVERN}) \end{bmatrix} \end{bmatrix}$$

---

(2) U-RULE: **integrateSysAsk**

PRE: $\begin{cases} \text{val}(\text{SHARED.LM}, \text{ask}(\text{sys},Q)) \\ \text{fst}(\text{PRIVATE.AGENDA}, \text{raise-question}(Q)) \end{cases}$

EFF: $\begin{cases} \text{push}(\text{SHARED.QUD}, Q) \\ \text{pop}(\text{PRIVATE.AGENDA}) \end{cases}$

(3) U-RULE: **integrateUserAnswer**

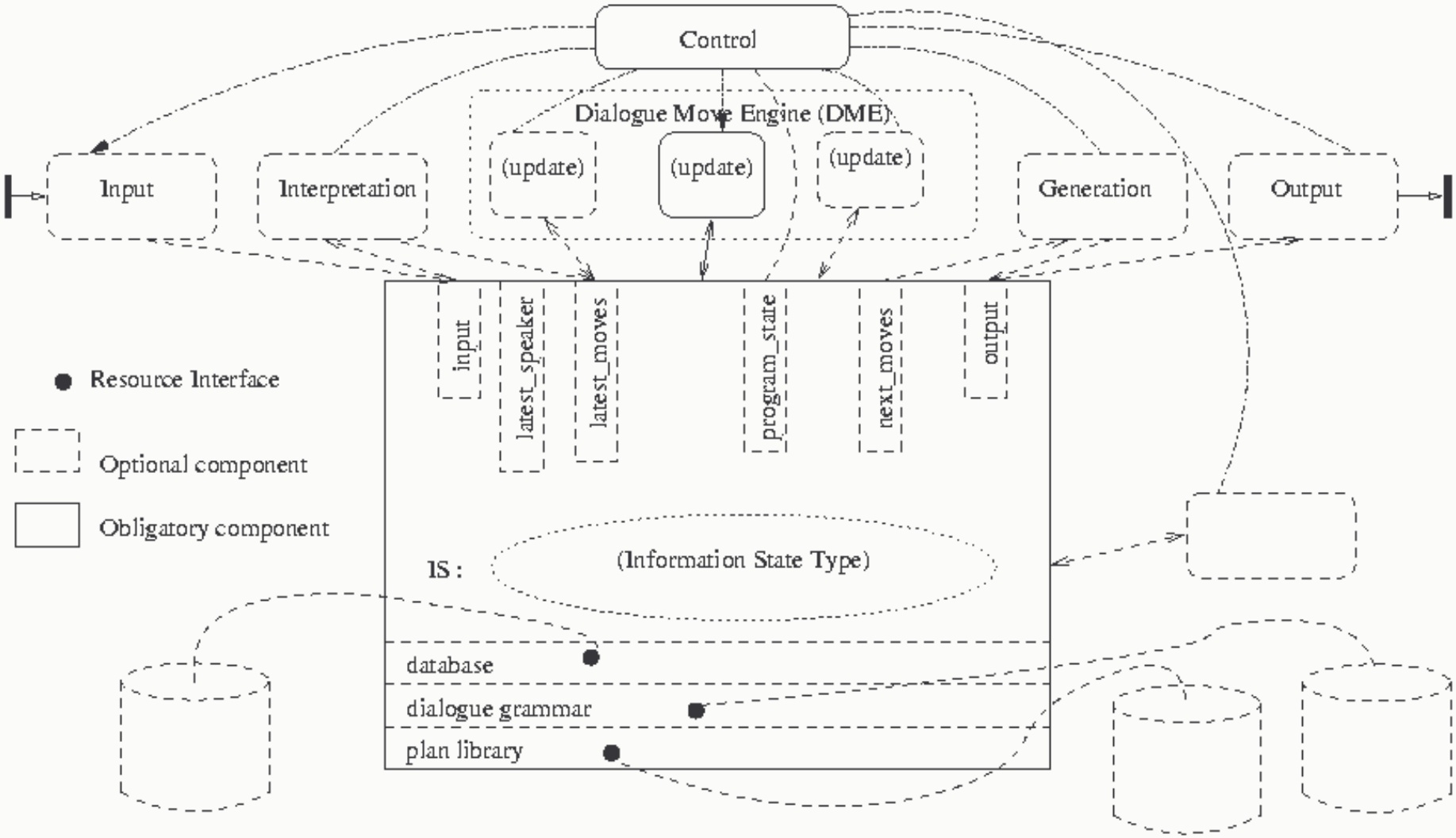
PRE: $\begin{cases} \text{val}(\text{SHARED.LM}, \text{answer}(\text{usr},A)), \\ \text{in}(\text{SHARED.QUD}, Q) \\ \text{answer-to}(\ A, Q\ ) \\ \text{apply}(\ Q, A, P\ ) \end{cases}$

EFF: $\begin{cases} \text{pop}(\text{SHARED.QUD}, Q), \\ \text{add}(\text{SHARED.BEL}, P) \end{cases}$

TRINDI Kit

- Dialogue Move Engine (**DME**) Toolkit
- Hauptfunktionen:
  - **Updaten** des Information state
  - basierend auf den **Bedingungen** der Dialogue Moves
  - Auswahl und Aufruf der Dialogue Moves
- DME übernimmt das **Dialog Management** vollständig
- zusätzliche Module/Funktionen
  - User Interface
  - Interpretation (z.B. Ermitteln des LM)
  - Generation
  - Control (z.B. parallel oder seriell)

TRINDI Kit - Architecture



Components : Obligatory / User-defined

- **Obligatorische** Komponenten:
  - Information State Type
  - DME module
  - Update Rules
  - Update alogorithm
  - Control Module
  - Control Algorithm
- **Benutzerdefinierte** Komponenten (angebunden durch Interface-Variablen):
  - Input modules
  - Interpreter
  - Generator

GoDIS

- Mit TRINDI Kit erstellt
- Einfache Algorithmen für Control, Update und Selection.
- Feature: question accomodation = User kann nicht gestellte, jedoch offensichtlich erscheinende Fragen beantworten.
- Aufbau:
  - 8 Dialogue Move Types: ASK, ANSWER, REPEAT, REQUEST_REPEAT, GREET, GOODBYE, THANK, QUIT
  - Information State geteilt in SHARED und PRIVATE

    Zusätzlich: PLAN (langfristige Ziele)

    TMP (bisher nicht bestätigte Annahmen)

GoDIS: Update rules

- **Besonderheit**: U-Rules „passen sich an“, falls die erforderlichen Bedingungen nicht in QUD oder PLAN zu finden sind
- **Beispiel**:

  J: what month do you want to go
  P: well around 3rd 4th april / some time there
  P: as cheap as possible

  angemessene Frage wäre:
  „What kind of price does P want for the ticket ?“
- Äußerung kann durch U-Rule aus dem Kontext abgeleitet werden

GoDIS: Information State Example

- Information State nach „what city do you want to go from ?“

```
Sys: welcome to the travel agency!
Usr: flights to paris
Sys: what city do you want to go from?
Usr: london
Sys: when do you want to travel?
Usr: in april, as cheap as possible
Sys: do you want a return ticket?
Usr: yes please
Sys: the price is $300
```

$$
\left[\begin{array}{lll}
\text{PRIVATE} & = & \left[\begin{array}{lll}
\text{BEL} & = & \{\} \\
\text{AGENDA} & = & \langle\rangle \\
\text{PLAN} & = & \left\langle\begin{array}{l}
\text{RAISE}(\text{R}\hat{}(\text{RETURN}=\text{R})), \\
\text{RAISE}(\text{M}\hat{}(\text{MONTH}=\text{M})), \\
\text{RAISE}(\text{C}\hat{}(\text{CLASS}=\text{C})), \\
\text{RESPOND}(\text{P}\hat{}(\text{PRICE}=\text{P}))
\end{array}\right\rangle \\
\text{TMP} & = & \text{(same as SHARED)}
\end{array}\right] \\
\text{SHARED} & = & \left[\begin{array}{lll}
\text{BEL} & = & \{(\text{TO}=\text{PARIS}),(\text{HOW}=\text{PLANE})\} \\
\text{QUD} & = & \langle\text{X}\hat{}(\text{FROM}=\text{X})\rangle \\
\text{LM} & = & \text{ASK}(\text{SYS},\text{Y}\hat{}(\text{FROM}=\text{Y}))
\end{array}\right]
\end{array}\right]
$$

GoDIS: accommodateQuestion U-Rule

$$
\left[\begin{array}{lll}
\text{PRIVATE} & = & \left[\begin{array}{lll}
\text{BEL} & = & \{\} \\
\text{AGENDA} & = & \langle\rangle \\
\text{PLAN} & = & \left\langle\begin{array}{l}
\text{RAISE}(\text{R}\hat{}(\text{RETURN}{=}\text{R})), \\
\text{RAISE}(\text{M}\hat{}(\text{MONTH}{=}\text{M})), \\
\text{RAISE}(\text{C}\hat{}(\text{CLASS}{=}\text{C})), \\
\text{RESPOND}(\text{P}\hat{}(\text{PRICE}{=}\text{P}))
\end{array}\right\rangle \\
\text{TMP} & = & (\text{same as SHARED})
\end{array}\right] \\
\text{SHARED} & = & \left[\begin{array}{lll}
\text{BEL} & = & \{(\text{TO}{=}\text{PARIS}), (\text{HOW}{=}\text{PLANE})\} \\
\text{QUD} & = & \langle \text{X}\hat{}(\text{FROM}{=}\text{X})\rangle \\
\text{LM} & = & \text{ASK}(\text{SYS}, \text{Y}\hat{}(\text{FROM}{=}\text{Y}))
\end{array}\right]
\end{array}\right]
$$

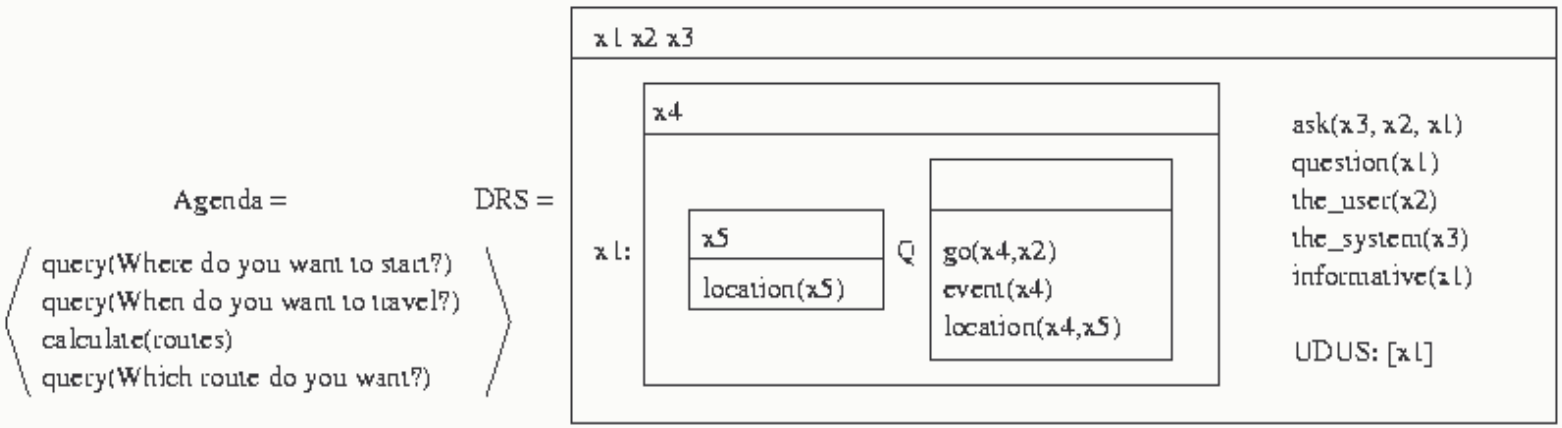
$$
\text{U-RULE: } \mathbf{accommodateQuestion}(Q, A)
$$

$$
\text{PRE: } \begin{cases}
\text{val}(\text{SHARED.LM}, \text{answer}(\text{usr}, A)), \\
\text{in}(\text{PRIVATE.PLAN}, \text{raise}(Q)) \\
\text{answer-to}(A, Q)
\end{cases}
$$

$$
\text{EFF: } \begin{cases}
\text{del}(\text{PRIVATE.PLAN}, \text{raise}(Q)) \\
\text{push}(\text{SHARED.QUD}, Q)
\end{cases}
$$

Andere TRINDIKit Systeme

- EDIS
- MIDAS: Benutzt DRS Strukturen der DRT (Kamp & Reyle) als Grundlage für den Information State



- SIRIDUS

TRINDI: Resultat

- Großer Schritt in Richtung „best practice“ bezüglich Dialogsystemen
- ToolKit zum Erstellen von Dialog-Managern
- Eng an der Theorie der Dialogverarbeitung
- Brücke zwischen „praktischen“ Dialogsystemen und eher theoretisch motivierten Ansätzen
- Bietet die Möglichkeit, gleichzeitig mehrere Aspekte der Dialogverarbeitung zu betrachten