State-based Systems

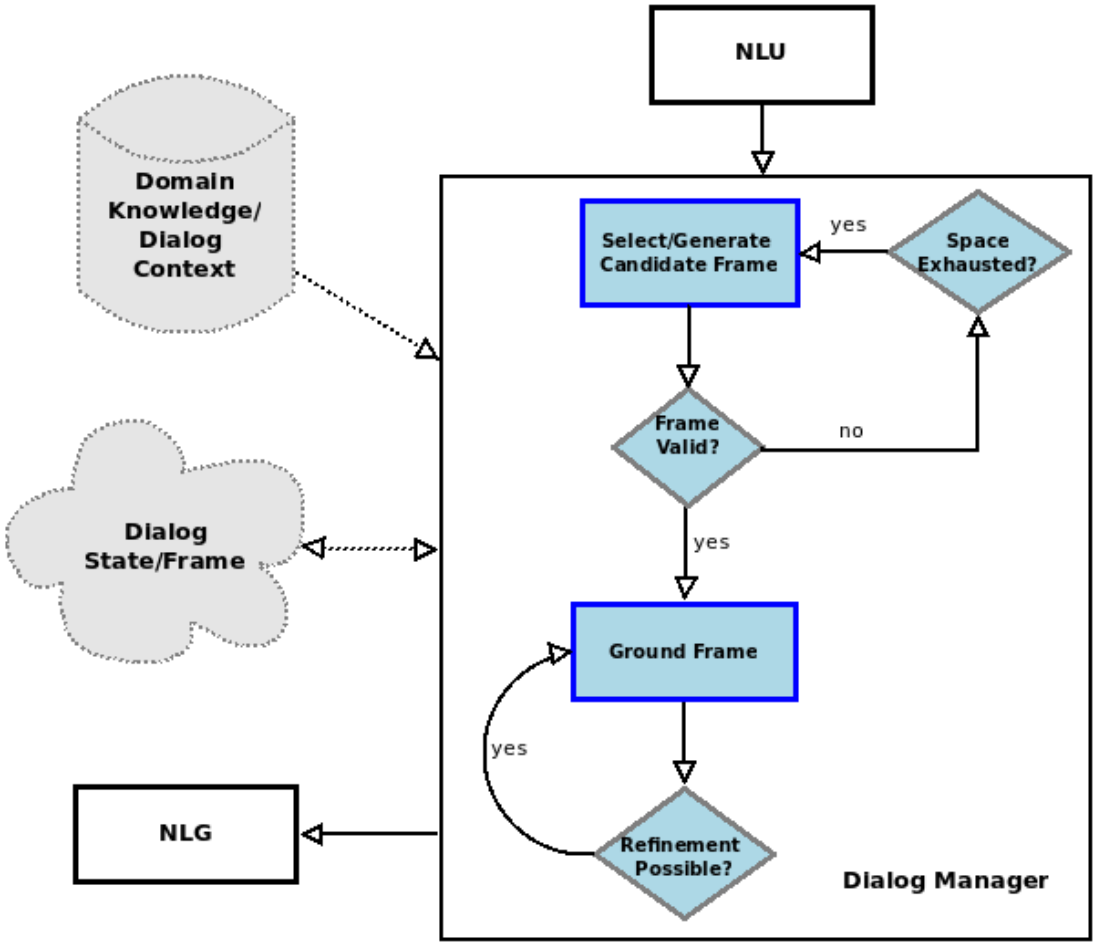
Camilo Thorne

DWS Group, Universität Mannheim, Germany
camilo@informatik.uni-mannheim.de

ESSLLI 2015
Barcelona, 10.8.2015-14.8.2015

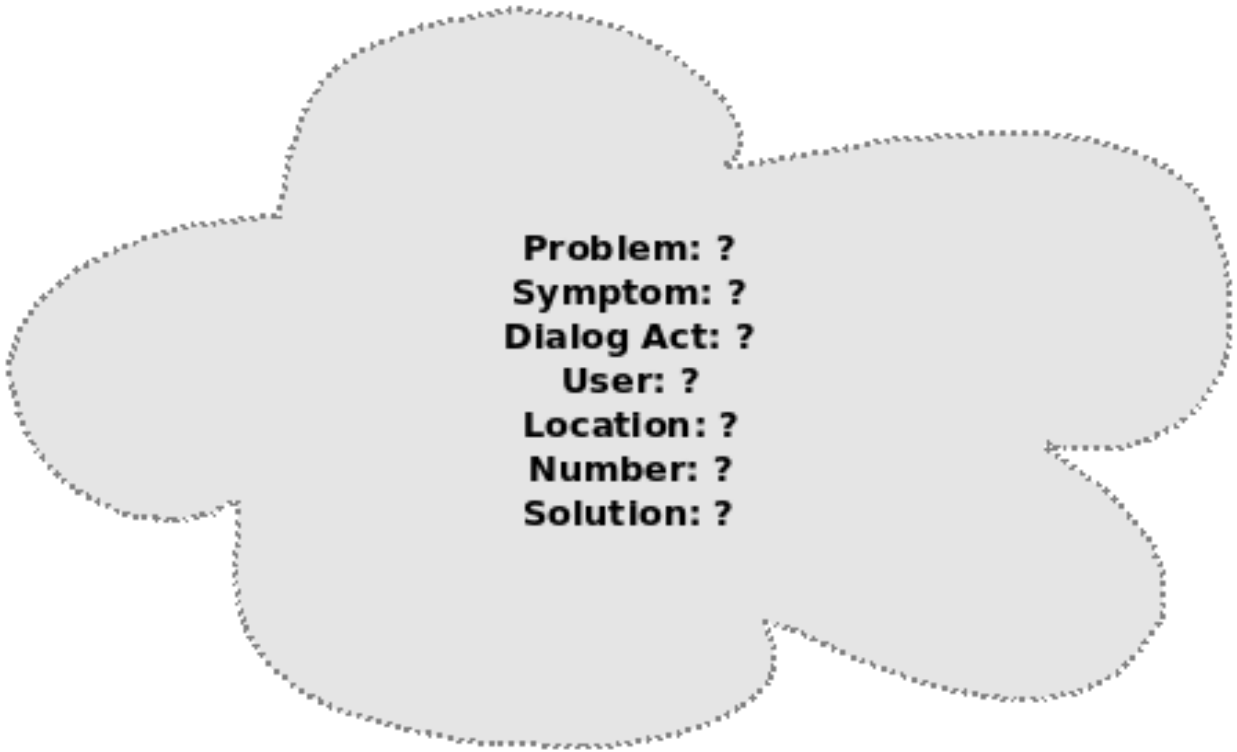
Outline

Reasoning Systems



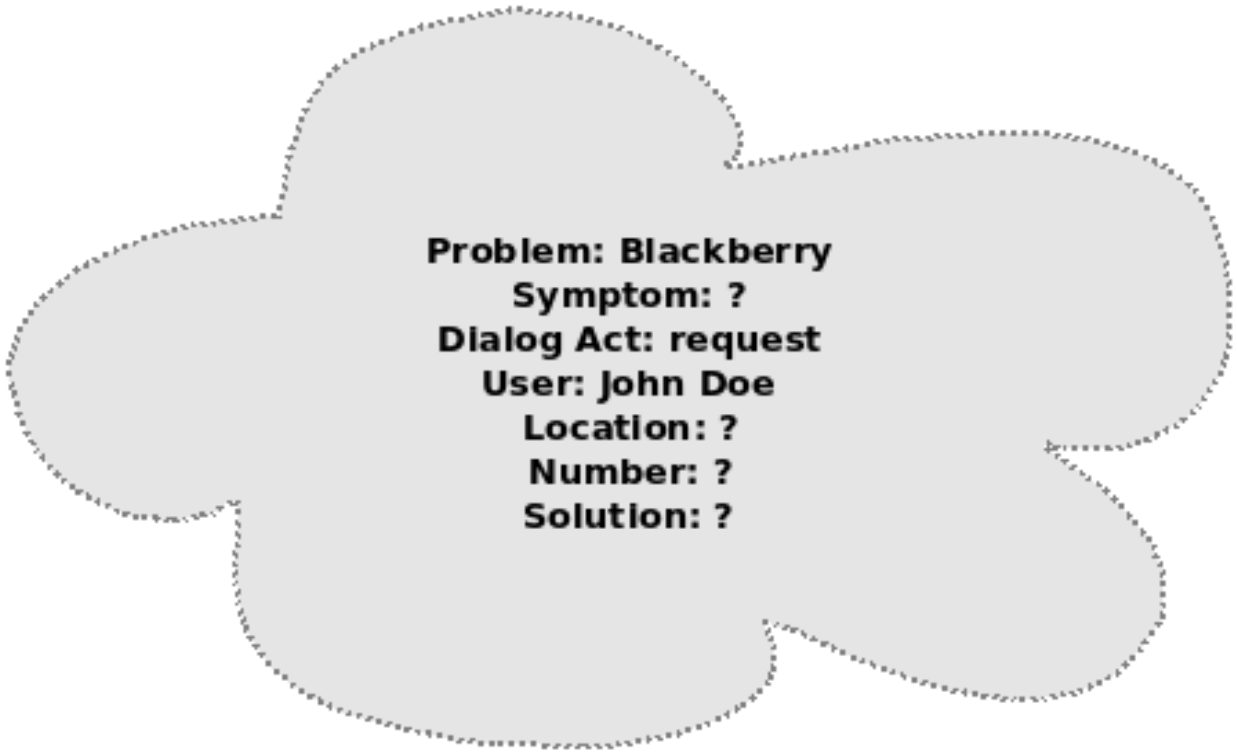
planner

Dialog States



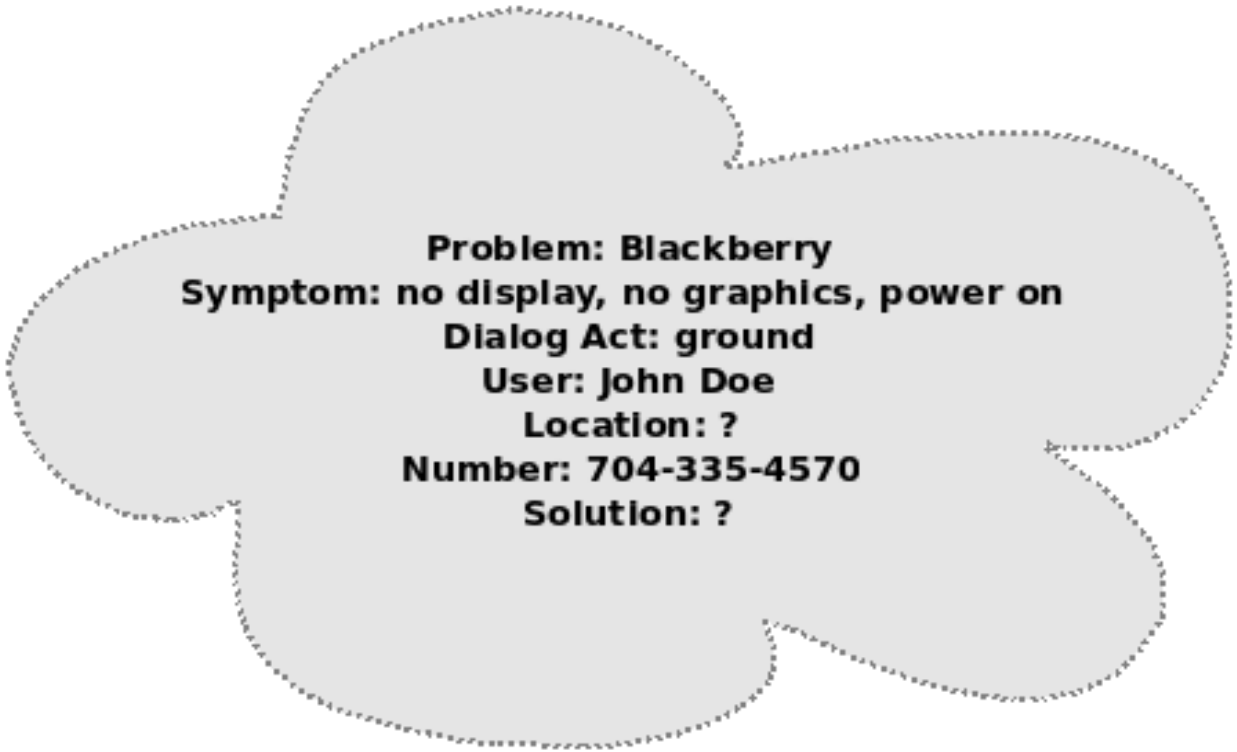
⇒ At the beginning, the frame/state is empty (greet)

Dialog States



$\Rightarrow$ The user request induces a candidate problem (request)

Dialog States

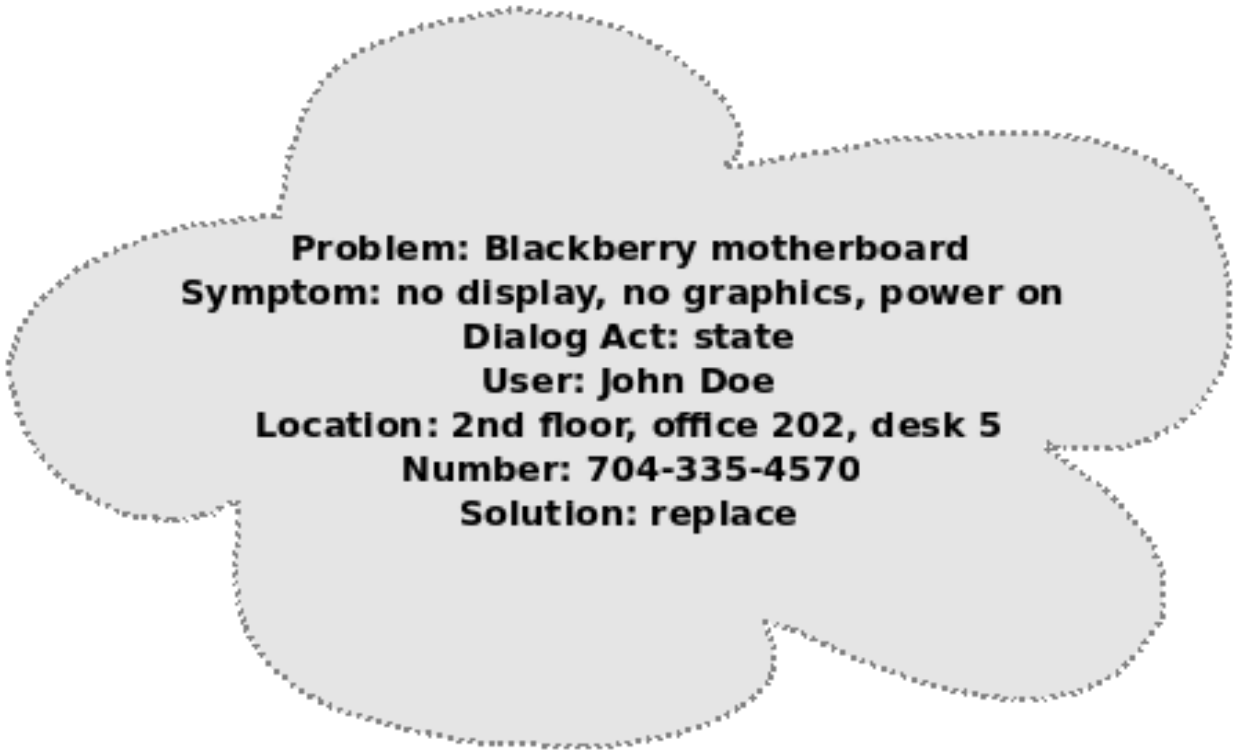


$\Rightarrow$ A conversation is engaged to gather more information (ground)

Dialog States

$\Rightarrow$ We continue until we fill the frame (ground)

Dialog States



$\Rightarrow$ A solution is determined and the dialog ends (state + greet)

Rule-based Management [JM09]

A rule is an expression of the form

```
GREET(u,p,a):
  PRE:
        User(u), Problem(p), DA(a), a=request
  POST:
        AddProblem(p), AddDA(a), AddUser(u),
        Reply("Thank you for choosing to
        chat with IT...")
```

such that if Boolean conditions

```
  User(u), Problem(p), DA(a), a=request
```

are satisfied, a number of actions are triggered

```
  KeepProblem(p), KeepDA(a), KeepUser(u),
  Reply("Thank you for choosing to chat with IT...")
```

Dialog State Update [JM09]

- Preconditions check for the current state S of the dialog
- Actions induce a change in S', giving rise to a new state S'

Dialog State Update [JM09]

- Preconditions check for the current state S of the dialog
- Actions induce a change in S', giving rise to a new state S'

- Rules can be seen a "function" that updates DS S to S'
- The update operation can be

1. deterministic vs. non-deterministic
2. symbolic vs. probabilistic

Dialog State Update [JM09]

- Preconditions check for the current state S of the dialog
- Actions induce a change in S', giving rise to a new state S'

- Rules can be seen a "function" that updates DS S to S'
- The update operation can be

  1. deterministic vs. non-deterministic
  2. symbolic vs. probabilistic

- Rules may update states independently or may depend on each other or exploit context $\Rightarrow$ reasoning & planning

Dialog State Update [JM09]

- Preconditions check for the current state S of the dialog
- Actions induce a change in S', giving rise to a new state S'

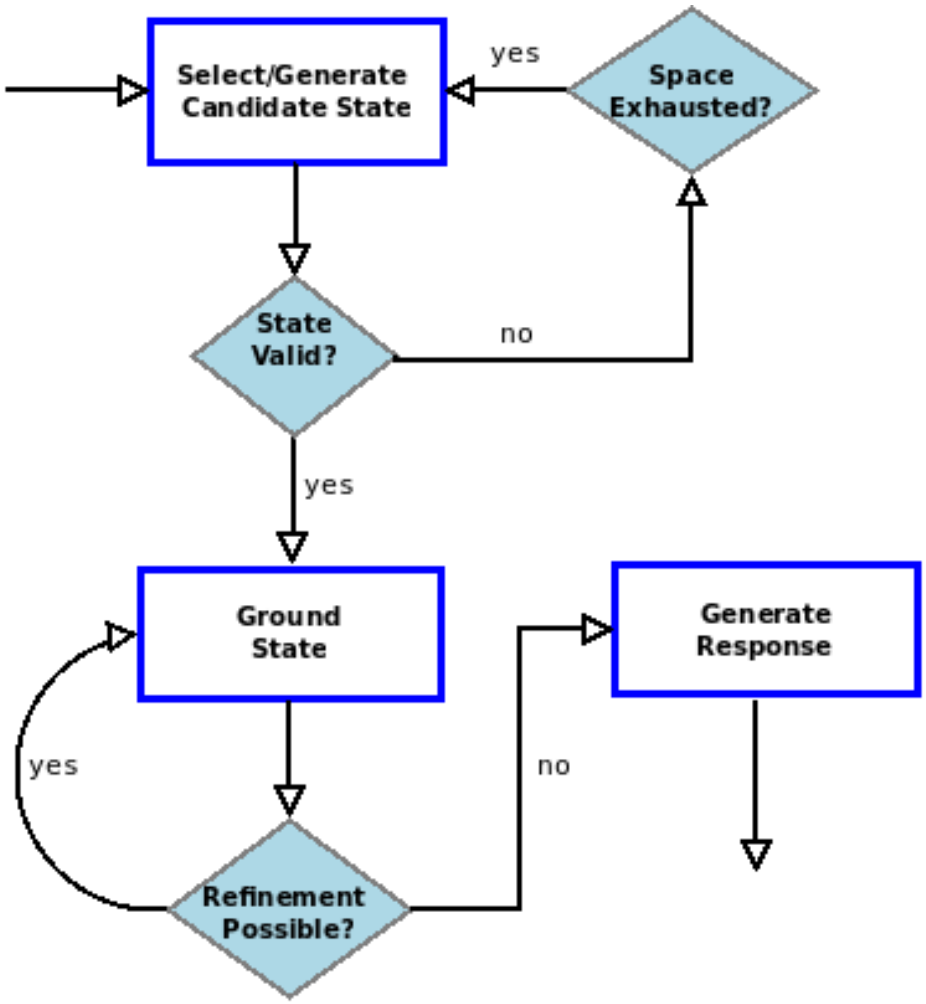
- Rules can be seen a "function" that updates DS S to S'
- The update operation can be

  1. deterministic vs. non-deterministic
  2. symbolic vs. probabilistic

- Rules may update states independently or may depend on each other or exploit context $\Rightarrow$ reasoning & planning

- Via side effects many more operations are possible

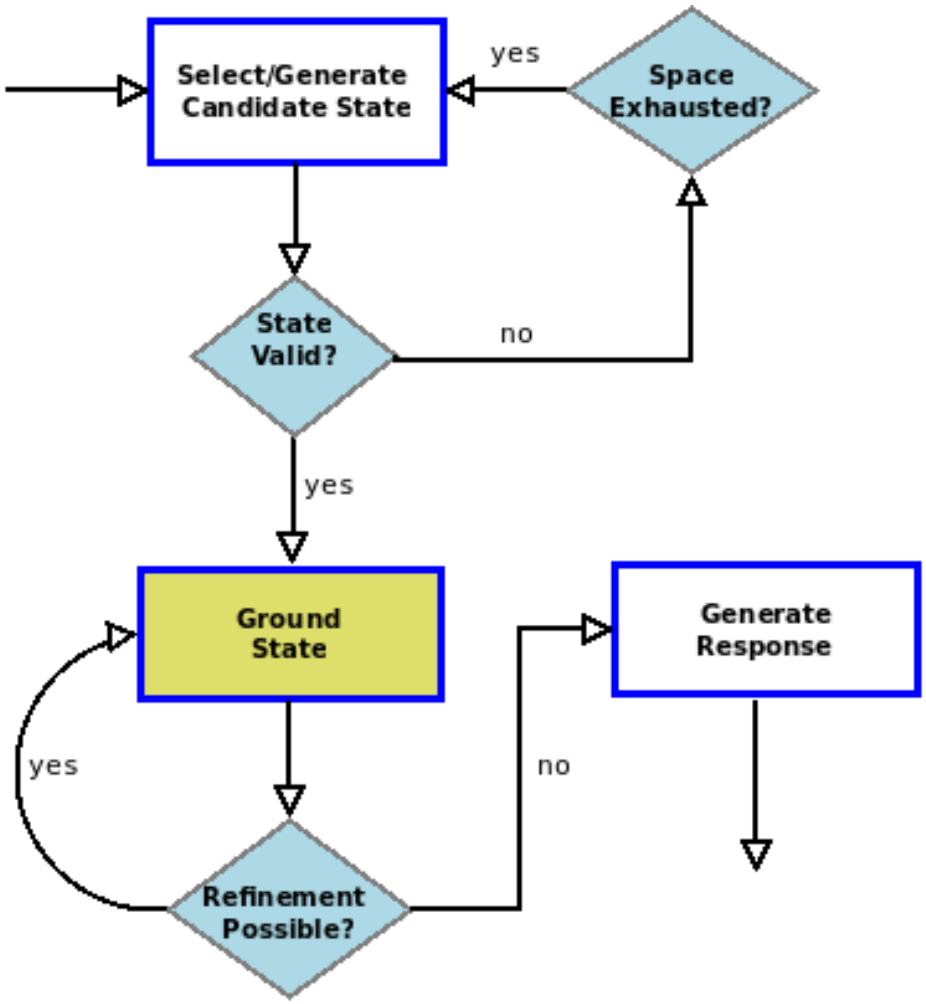
Dialog State Update (ctd.)



More in general, rule-based updates are iteratively executed until a complete DS, the goal state is reached

The grounding module is usually responsible for this task

Dialog State Update (ctd.)



More in general, rule-based updates are iteratively executed until a complete DS, the goal state is reached

The grounding module is usually responsible for this task

Examples of State-based Systems

System	Internal State	Manager Type	Domain	Domain Context	Initiative	Anaphora
OpenDial [Lis15]	yes	rule-based/ probabilistic	several	no	double	?
AIDA [BJK+13]	yes	rule-based	several	DB	?	?
Roundtable [FKM+13]	yes	rule-based	several		?	?
Boye et al. [Boy07]	yes	planning	technical	KB	?	?
Chakrabarti et al. [CL12]	yes	rule-based	technical	KB	system	?

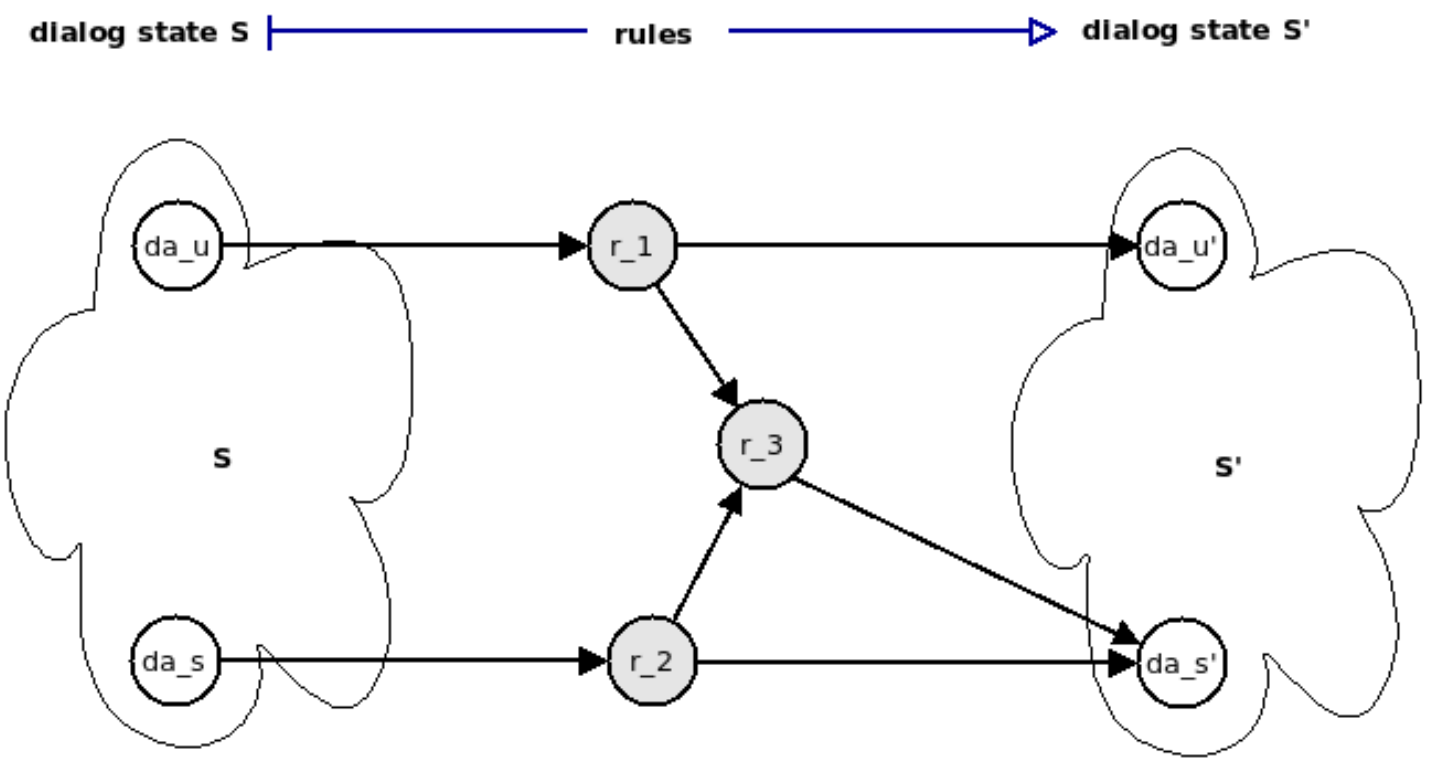
⇒ In what follows, we will illustrate the theory using P. Lison's OpenDial system

Examples of State-based Systems

System	Internal State	Manager Type	Domain	Domain Context	Initiative	Anaphora
OpenDial [Lis15]	yes	rule-based/ probabilistic	several	no	double	?
AIDA [BJK+13]	yes	rule-based	several	DB	?	?
Roundtable [FKM+13]	yes	rule-based	several		?	?
Boye et al. [Boy07]	yes	planning	technical	KB	?	?
Chakrabarti et al. [CL12]	yes	rule-based	technical	KB	system	?

⇒ In what follows, we will illustrate the theory using P. Lison's OpenDial system

Graph-based Representation [Lis15]

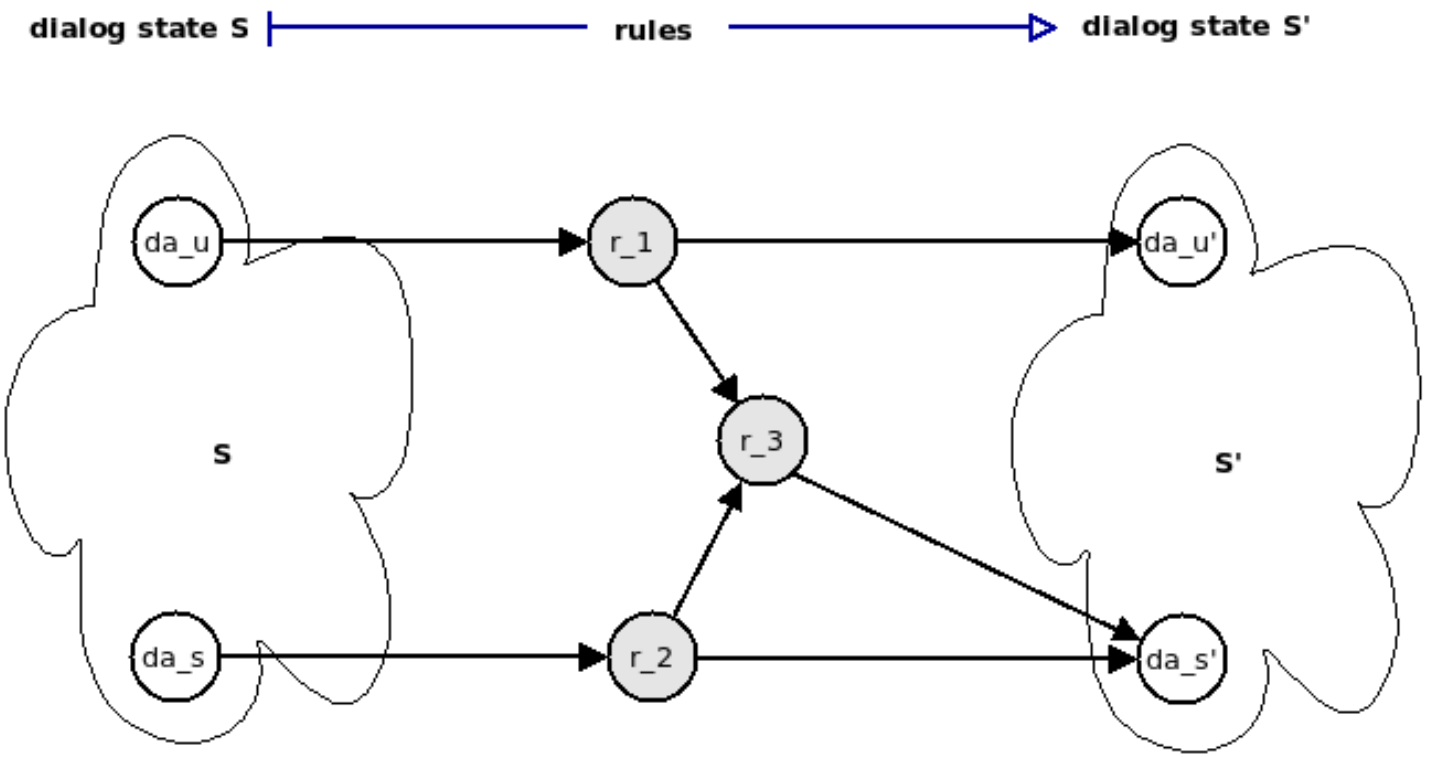


⇒ nodes denote rules and DAs

⇒ incoming edges denote preconditions

⇒ outgoing edges denote actions

Graph-based Representation [Lis15]



⇒ nodes denote rules and DAs

⇒ incoming edges denote preconditions Bayesian network!

⇒ outgoing edges denote actions

Beliefs, Desires & Intentions

- A planning dialog system can
  1. decide a course of action ⇒ intention
  2. based on its communicative goal/DA ⇒ desire
  3. and background knowledge/context/DS ⇒ belief

- At each step i the DM will
  1. observe the current state S_i of the DS
  2. access the background knowledge (context)
  3. set itself a goal expressed as a system DA
  4. do forward chaining of rules until either the goal or failure is reached

Remark: it is possible to implement a DM using a planning blackbox and dialog rules using rule languages such as the Planning Domain Definition Language (PDDL)

Probabilistic Dialog Management

Probablistic dialog management is based on Markov decision procedures (MDPs) defined at time (= turn) t

- Given

  1. S a sequence $s_0, \ldots, s_t$ of states (= history of DSs at t)
  2. A a sequence $a_0, \ldots, a_t$ of actions (= history of DM actions at t)
  3. transition assignment $b : S \times A \times S \rightarrow [1, 0]$ s.t.

$$b(s_i, a_i, s_{i+1}) = P(S = s_{i+1} \mid S = s_i, A = a_i)$$

  4. $r : S \times A \times S \rightarrow \mathbb{R}$ a reward assignment

- Find the policy $\pi : S \rightarrow A$ that maximizes expected cumulative reward:

$$\pi^* = \arg\max_{\pi} E[\sum_{i=0}^{t} r(s_i, a_i, s_{i+1}) \mid \pi]$$

$\Rightarrow$ If partially observable: add a sequence $O = o_0, \ldots, o_t$ of observations and a belief state b and use them to define r and π

A Simple Exercise

user:	blackberry problem
rep.:	what kind of problem?
user:	black screen
rep.:	anything else?
user:	keyboard unresponsive
rep.:	I'll pass you to the technician
user:	thanks!
rep.:	bye

EXO: How do we implement this Dialog in OpenDial?

Conclusions

1. Reasoning-based dialog systems incorporate rules, a DS and possibly background knowledge
2. They are in principle better at dealing with grounding than chatterbots
3. Their DM can be extended into a full-scale planner and support side-effects
4. Rules can incorporate probabilities to model ambiguity and uncertainty
5. But: most implementations understand only the vocabulary of their domain


Thank You!!!

References I

Rafael E. Banks, Ridong Jiang, Seokhwan Kim, Arthur Niswar, and Kheng Hui Yeo.

AIDA: Artificial intelligent dialogue agent.

In *Proceedings of SIGDIAL 2013*, 2013.

Johan Boye.

Dialogue management for automatic troubleshooting and other problem-solving applications.

In *Proceedings of the 8th SIGDial Workshop on Discourse and Dialogue*, 2007.

Chayan Chakrabarti and George F. Luger.

A semantic architecture for artificial conversations.

In *Soft Computing and Intelligent Systems and 13th International Symposium on Advanced Intelligent Systems (SCIS-ISIS 2012)*, 2012.

Eric Forbell, Nicolai Kalisch, Fabrizio Morbini, Kelly Christoffersen, Kenji Sagae, David Traum, and Albert A. Rizzo.

Roundtable: An online framework for building web-based conversational agents.

In *Proceedings of SIGDIAL 2013*, 2013.

References II

James R. Glass.

Challenges for spoken dialogue systems.

In *Proceedings of the 1999 IEEE ASRU Workshop*, 1999.

Daniel Jurafsky and James Martin.

Speech and Language Processing.

Prentice Hall, 2nd edition, 2009.

Pierre Lison.

A hybrid approach to dialogue management based on probabilistic rules.

Computer Speech & Language, 34(1):232–255, 2015.

Bayan Abu Shawar and Eric Atwell.

Different measurement metrics to evaluate a chatbot system.

In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, 2007.

References III

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella.

PARADISE: A framework for evaluating spoken dialogue agents.

In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics EACL 1997*, 1997.

Appendix - Computing π^*

To compute π^* it is sufficient to solve the Bellman recurrence

$$V(s) := \begin{cases} 0 & \text{if } s = s_0 \\ \sum_i b(s_i, \pi^*(s_i), s_{i+1}) \times [r(s_i, \pi^*(s_i), s_{i+1}) + V(s_{i+1})] & \text{otherwise} \end{cases} \quad (1)$$

for every state s, where

- $V(s)$ is the cum. reward ("utility") at state s
- $\pi^*(s) := \arg\max_a \Phi$ (Φ is the body of equation 1)

$\Rightarrow$ It can be solved efficiently using dynamic programming