Issue Based Dialogue Management, part 1

Staffan Larsson
Dialogue Systems II
Master in Language Technology programme
University of Gothenburg

September 18, 2018

Outline

Why teach Issue Based Dialogue Management?

- Compares favourably to the state of the art in dialogue management
- Not a black box - we know how it works
- Based on theories about dialogue, from linguistics and related subjects, rather than seeing dialogue management as a technical problem only
- Connection to local industry (Talkamatic)
- Expertise available (Talkamatic)

But it's so old!?

What has happened since 2002?

- Most existing commercial dialogue managers are similar to VoiceXML in performance
  - (probably) form-based
  - but with some degree of multimodality added
  - and connected to more or less sophisticated user models
- Academic research in the last 10 years or so has built on the information state approach but focused on using statistical methods to automatically learn some parts of dialogue management
- However, statistical models for dialogue management are (still) not used commercially
  - There have been some attempts but none so far successful
  - Hybrid systems more likely to succeed than purely statistical systems, but little work done so far in this direction
- If you go work with dialogue systems after graduating, you are likely to be working with a rule-based system

What has happened to IBiS/TDM since 2002?

- Issue-based dialogue management is being commercialised (Talkamatic AB, started 2008)
- IBiS/GoDiS system originally implemented in Prolog (logic programming) has been reimplemented in Python
- Multimodality has been added in the form of *multimodal menu-based dialogue*
- Android client available, system runs on server
- Uses a variety of speech recognition and speech synthesis systems, including Google ASR and Nuance ASR open recognizers
- NLU uses RASA.AI item NLG uses Grammatical Framework (GF), but uses simplified grammar format
- App development now done using XML; web based GUI development tool will developed in 2019

Some features of IBDM in IBiS1 (and TDM)

- Dialogue logic separated from domain knowledge
  - When building new app, only domain knowledge is added
  - Enables fast development and debugging
- Flexible dialogue managent, including
  - Question Accommodation enabling
    - Over-answering, other-answering
    - Intention recognition (limited in competitors)
  - Free switching between multiple active conversational topics without information loss (not handled by most competitors)
- Multimodal menu-based dialogue (not handled by competitors)
  - Allows switching freely between modalities at any time
  - Enabling eyes-free interaction (even without speech input)
- IBDM is intended as a generic theory of dialogue, and should be extendable to more complex interaction types (remains to be shown)

The aim of Larsson (2002)

- Explore *issue-based dialogue management*, an approach to dialogue management and dialogue modelling which regards issues, modelled semantically as questions, as a primary organizing and motivating force in dialogue
- We will do this in part by implementing our theory in a dialogue system
  - this will force our analysis to be precise
  - allow us to show how the different components of the theory interact to produce concrete behaviour when the system participates in a dialogue.
- Will proceed both on a theoretical and a practical implementation level
- Starting from a basic account of issue-based dialogue management, we gradually extend the coverage of the theory and the implementation to more complex types of dialogue.

Why study dialogue modelling and dialogue management?

Theoretical purpose:

- Provide models allowing us to explore how language, and especially spoken dialogue, is used in different activities
- What enables agents (human or machines) to participate in dialogue?
- What kind of information does a dialogue participant (or *DP*) need to keep track of?
- How is this information used for interpreting and generating linguistic behaviour?
- How is dialogue structured, and how can these structures be explained?

Why study dialogue modelling and dialogue management?

Practical purpose:

- Building dialogue systems to enable natural human-computer interaction.
- There is a widely held belief that interfaces using spoken dialogue may be the "next big thing" in the field of human-computer interaction
- Before this can happen, dialogue systems must become more flexible than currently available commercial systems
- To achieve this, we need to base our implementations on reasonable theories of dialogue modelling and dialogue management.

The implementation of dialogue systems can also feed back into the theoretical modelling of dialogue, provided the actual implementations are closely related to the underlying theory of dialogue.

Outline

Information State update approach

- View of dialogue management functions in terms of information state
- Key to this approach:
  - Identifying the relevant aspects of information in dialogue
  - How this information is updated as the dialogue proceeds

Information State

- The information state of a dialogue represents
  - cumulative additions from previous actions in the dialogue and motivating future action
  - the information that an agent needs to keep track of in order to be able to participate in a dialogue
  - Example: statements generally add propositional information
  - Example: questions generally provide motivation for others to provide specific statements
- Related concepts:
  - common ground, mutual knowledge
  - conversational scoreboard, dialogue gameboard, dialogue state
  - mental state

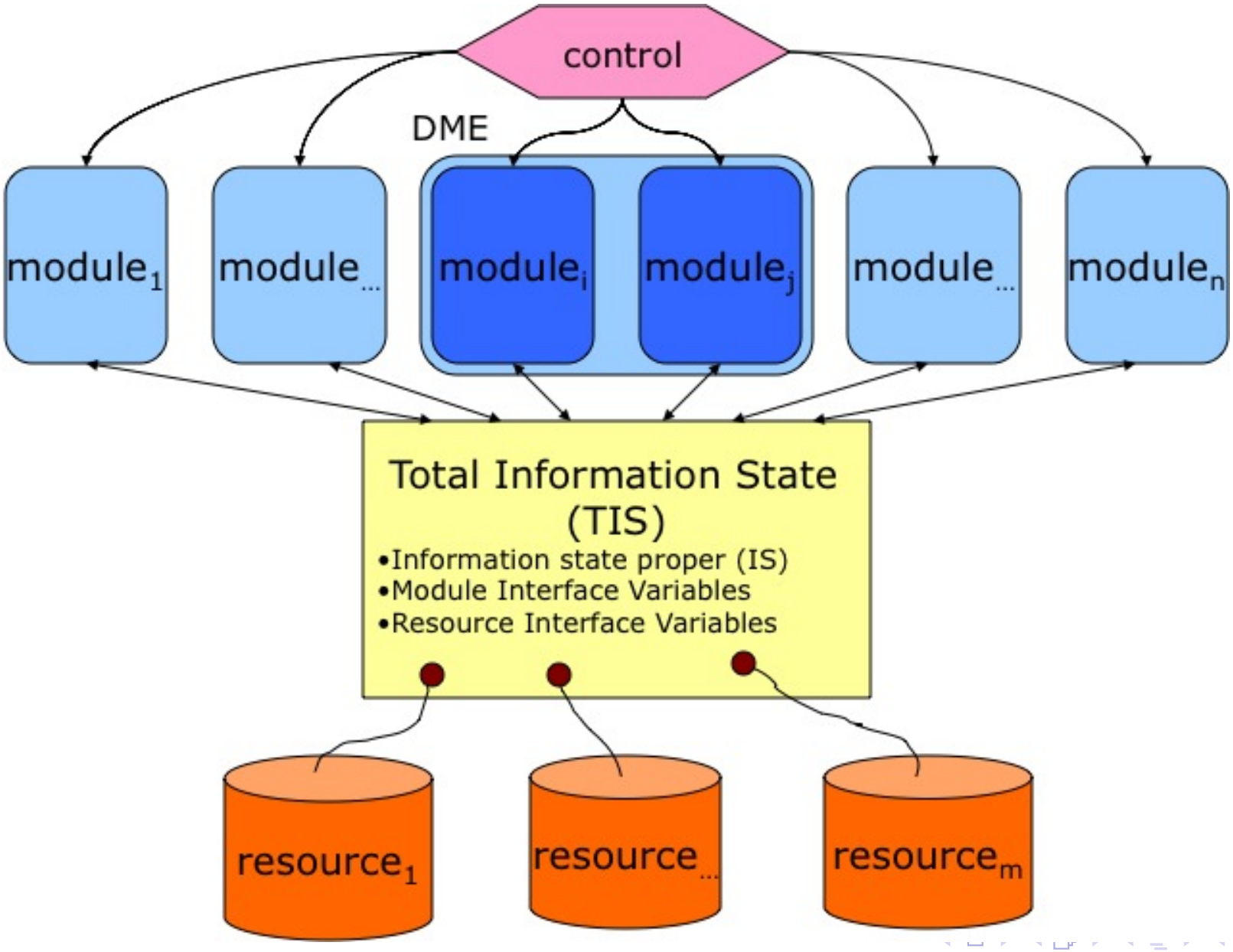
Components of an ISU theory of dialogue modeling

- A description of the informational components of the theory of dialogue modeling
  - linguistic structure
  - obligations, commitments (social attitudes)
  - questions under discussion
  - beliefs, mutual beliefs
  - intentions, intentional structure, dialogue plans, agenda
  - user models
  - ...
- *Information state*: a formal representation of informational components, e.g.
  - lists, sets, stacks
  - typed feature structures
  - records, record types
  - Discourse Representation Structures (DRSs)
  - propositions, possible worlds
  - ...

Components of an ISU theory of dialogue modeling

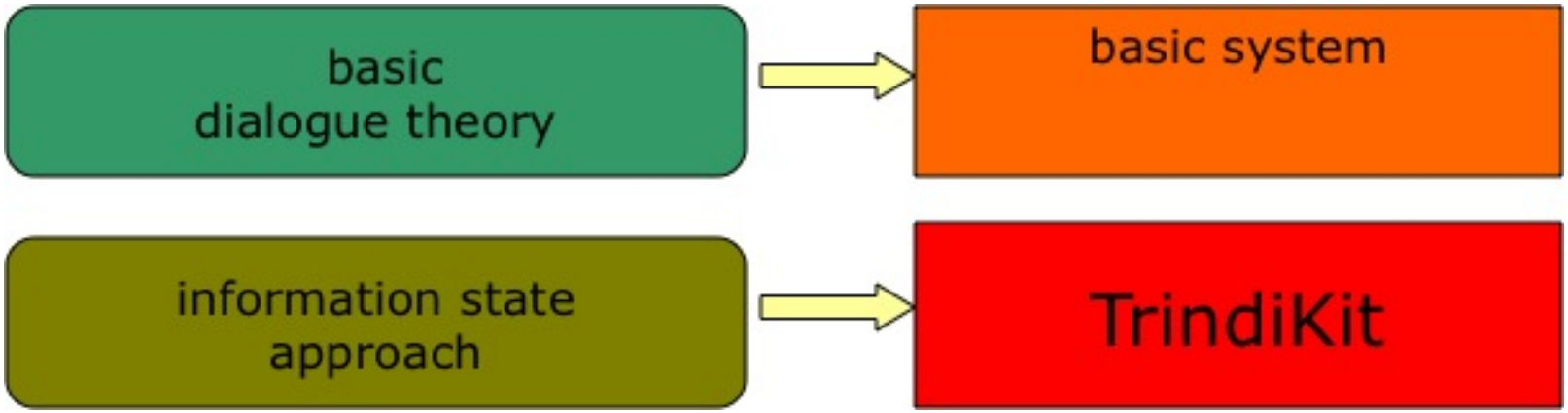
- A set of *dialogue moves*
  - trigger the update of the information state
  - correlated with externally performed actions such as particular natural language utterances
- A set of *update rules*
  - govern updating of the information state given various conditions of the current information state and performed dialogue moves
- A set of *selection rules*
  - license choosing a particular dialogue move to perform given conditions of the current information state
  - enabling participating in a dialogue rather than just monitoring one
- An *update strategy* for deciding which rules to select at a given point from the set of applicable ones
  - picking the first rule that applies
  - more sophisticated arbitration mechanisms based on game theory, utility theory or statistical methods

TrindiKit architecture



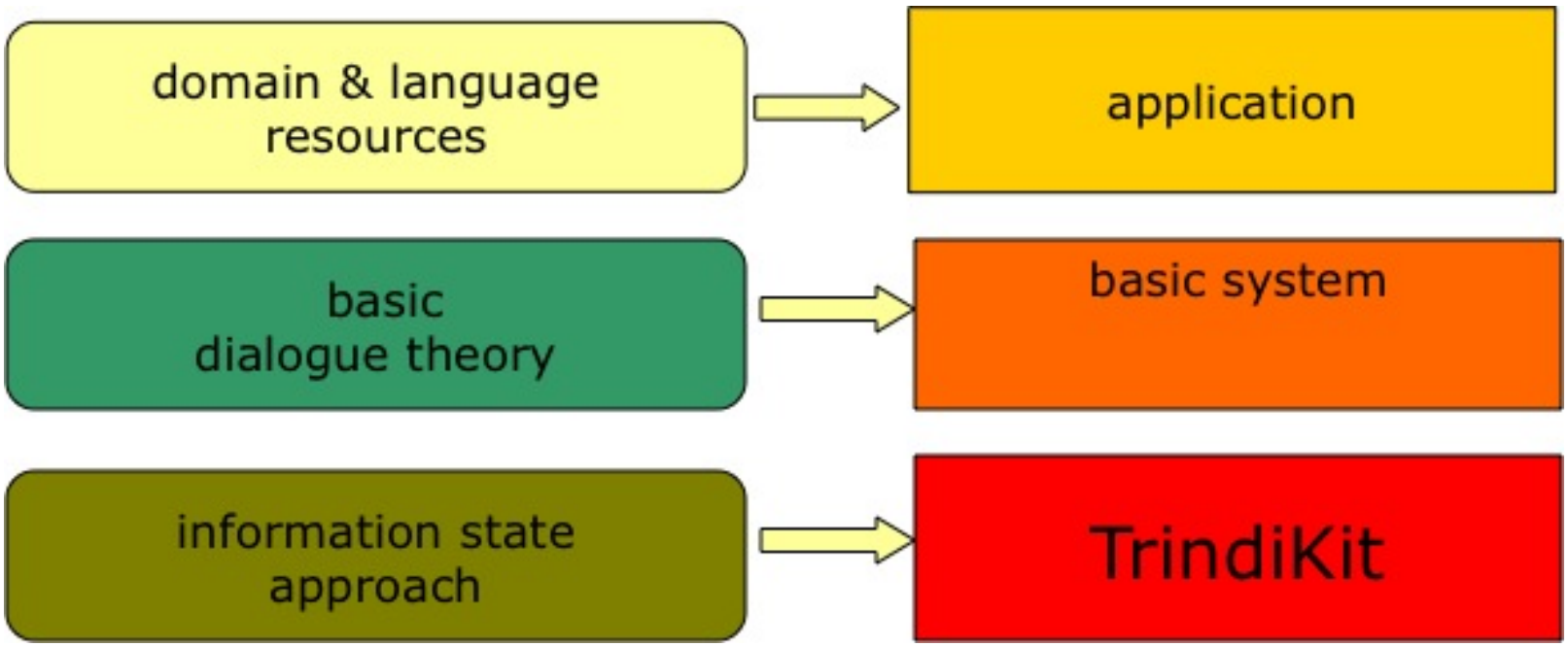
How to build a basic system

- Formulate a basic dialogue theory
  - Information state
  - Dialogue moves
  - Update rules
- Add appropriate modules (speech recognition etc)



How to build an application

- Add application-specific resources



Dialogue Move Engine (DME)

- module or group of modules responsible for
  - updating the IS based on observed moves
  - selecting moves to be performed
    - (MDPs is a special kind of selection module which uses statistics to select next move)
- dialogue moves are associated with IS updates using IS update rules
- update rules: rules for updating the TIS
  - rule name and class
  - preconditon list: conditions on TIS
  - effect list: updates to TIS

example IS and rule

- Information state type:
$$\begin{bmatrix} \text{BEL} & : & \text{Set(Prop)} \\ \text{DES} & : & \text{Set(Prop)} \\ \text{INT} & : & \text{Set(Action)} \\ \text{MBEL} & : & \text{Set(Prop)} \\ \text{LM} & : & \text{Set(Move)} \end{bmatrix}$$
- Moves: assert(P), askif(P)
- Sample rule:
  - integrate_assert
  - if
    - in(lm, assert(P))
  - then
    - add(is.mbel, P)
    - add(is.bel, P)
    - del(lm, assert(P))

Example rule application

IS = [
BEL = { happy(sys) }
DES = { knowif(happy(usr)) }
INT = { }
MBEL = { }
]

LM = { assert(happy(usr) }

Rule application: integrate_assert,
> add(IS/MBEL, happy(usr)),
> add(IS/BEL, happy(usr)),
> del(LM, assert(happy(usr))

IS = [
BEL = { happy(sys), happy(usr) }
DES = { knowif(happy(usr)) }
INT = { }
MBEL = { happy(usr) }
]

LM = { }

Outline

Disadvantages of finite state approach

- Dialogues are scripted utterance by utterance on a very concrete level
- Each utterance leads to a new state, where various possible followup utterances are allowed, each leading to a new state
- As a model of dialogue this approach is problematic since it does not really explain dialogue structure; rather, it describes it, and in a very rigid way
- As a tool for dialogue modelling, the finite state approach is in practice limited to very simple dialogue where the number of available options at any point in the dialogue is very small.

Disadvantages of form-based approach

- Dialogue is reduced to the process of filling in a form.
- The form-based approach is used e.g., in VoiceXML
- Forms provide a very basic formalism which is sufficient for simple dialogue, but it is hard to see how it can be extended to handle more complex kinds of dialogue, e.g., negotiative, tutorial, or collaborative planning dialogue.
- Issue-based dialogue management, on the other hand, is (in principle) independent of the choice of semantic formalism.
- This enables an issue-based system to be incrementally extended to handle dialogue phenomena involving more complex semantics.

Advantages of using questions to model dialogue

- In an abstract sense, the goal of all practical dialogue is to communicate information which is useful in some activity.
- This means that conversational goals should describe missing information.
- To fulfil a conversational goal, what we need to do is to communicate the missing information.
- Issues, or questions, essentially are entities specifying certain pieces of as yet unavailable information.
- That is, conversational goals can to a large extent be modelled as questions.

Advantages of using questions to model dialogue, cont'd

- In addition to issues arising from the activity in which a dialogue takes place, there are also "meta-issues" which arise from the dialogue itself.
  - Did the other participant understand my utterance correctly?
  - Should I accept what she just said as true?
  - Have we reached a mutual understanding of what I meant with my previous utterance?

Outline

Outline

Ginzburg's Dialogue Gameboard

- Ginzburg's DGB provides a rich structure which includes propositions, questions, and dialogue moves
- Previous accounts (Stalnaker, Lewis) were mainly interested in how *assertions* change the common ground
- Ginzburg's inclusion of questions enables the modelling of how the raising of questions affect the common ground.

Ginzburg's Dialogue Gameboard, cont'd

Ginzburg structures a participant's version of the DGB into three separate fields:

- FACTS: set of commonly agreed upon facts
- QUD ('questions under discussion'): a set that specifies the currently discussable questions, partially ordered by $\prec$ ('takes conversational precedence').
  - If q is maximal in QUD, it is permissible to provide any information specific to q using (optionally) a short answer.
- LATEST-MOVE: content of the *latest move* made: it is permissible to make whatever moves are available as reactions to the latest move.

QUD

- QUD is intended to model a view of conversation as the setting up of possible questions to discuss and the subsequent resolving of some of these questions.
- At any time, a speaker may choose to add something to the QUD, or to address one of the questions in QUD.
- The effect on the DGB of a DP asking a question is to
  a. "significantly restrict the space of felicitous follow-up assertions or queries", and
  b. "to license an elliptical form which (overtly) conveys only the focus component of the response"

QUD, cont'd

- Example
  - *S*'s utterance below results in the question "What city does the user want to go to?" being added to QUD.
  - This licenses the elliptical response in *U*'s utterance

S> `What city do you want to go to?`

U> `paris`

DGB and Information State

- Ginzburg also stresses that the DGB is a *quasi-shared* object
  - each DP has her own version of the DGB and there may be differences (mismatches) between the DGBs of different DPs.
  - This follows from the view that DGB and UNPUB-MS are components of a DPs mental state.
- A DP's information state may include information that she assumes to be *shared* with other DPs
  - Each dialogue participant (DP) has her own take on the shared state
  - DP's strive to coordinate their takes on the shared state ("grounding")
  - Ideally, the shared state is in fact shared (all DPs have the same take on the dialogue state)
- Information state needs only include dynamic information (i.e. information that may be updated during the dialogue)
- An agent may also keep track of *private* information, such as private beliefs and intentions/plans

Outline

Overview of IBiS1

- Our initial system will be able to handle simple inquiry-oriented dialogue,
- Inquiry Oriented Dialogue, or *IOD*:
  - any dialogue whose sole purpose is the transference of information
  - and which does not involve any DP (Dialogue Participant) assuming (or trying to make another DP assume) commitments or obligations concerning any non-communicative actions outside the dialogue
- Our account of basic issue-based dialogue management will be based on Ginzburg's theory.

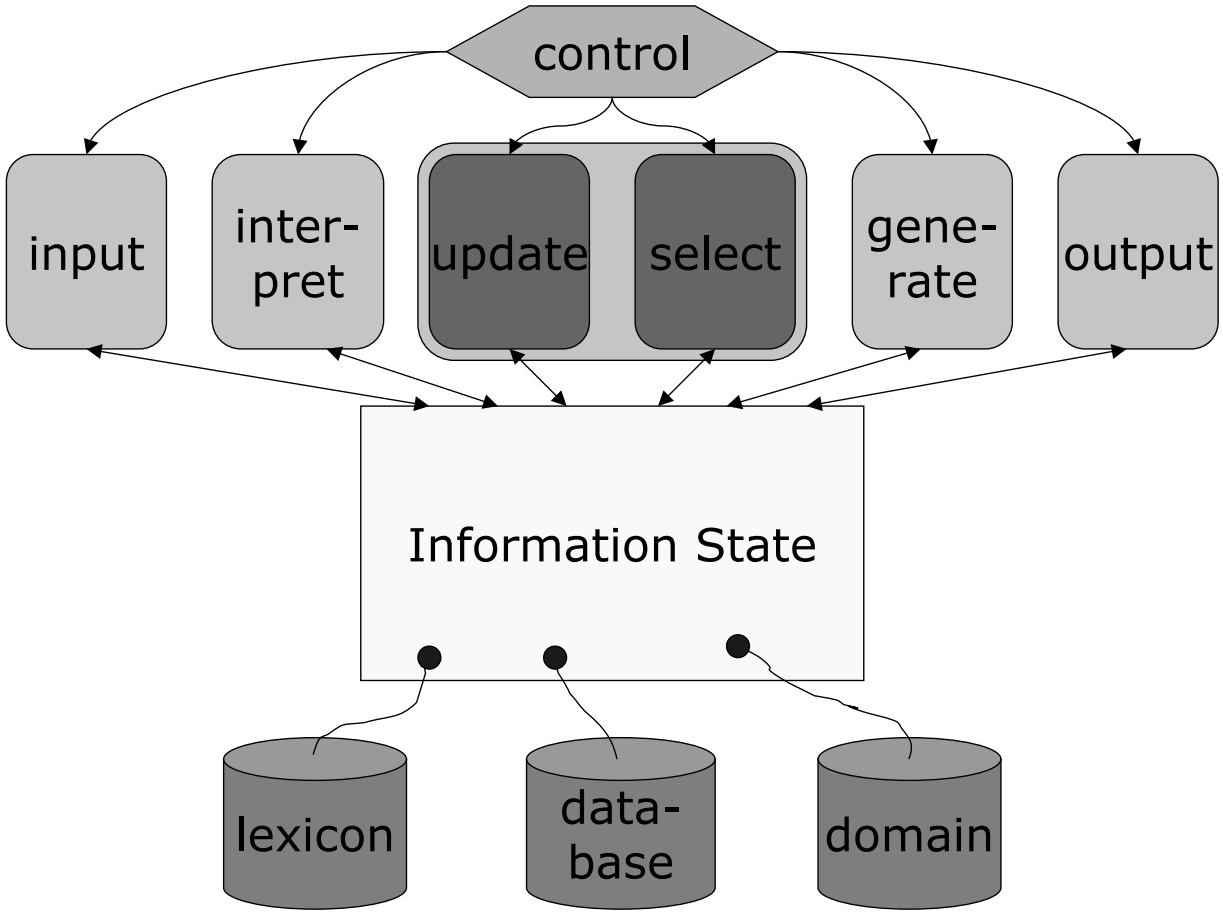
A sample inquiry-oriented dialogue

S> Welcome to the travel agency!
U> hello
U> price information please
S> How do you want to travel?
U> a flight please
S> What city do you want to go to?
U> paris
S> What city do you want to go from?
U> from london
S> What month do you want to leave?
U> in april
S> What day do you want to leave?
U> as cheap as possible
S> What day do you want to leave?
U> the fifth
S> What class did you have in mind?
U> as cheap as possible
S> It will cost 123 crowns.
U> bye

S> Thank you for your visit!

Outline

IBiS1 architecture



IBiS1 architecture components

- the Information State (IS)
- domain-independent modules: **input**, **interpret**, **generate** and **output**
- the Dialogue Move Engine (DME), consisting of two modules (**update** and **select**); the DME is responsible for updating the IS based on observed moves, and selecting moves to be performed by the system.
- a controller, wiring together the other modules, either in sequence or through an asynchronous mechanism.
- three domain-dependent resources: Database, Lexicon, and Domain Knowledge
- In TDM, these correspond to device, grammar, domain and ontology

IBiS1 Control algorithm

repeat ⟨ **select**
if not is_empty($NEXT_MOVES)
then ⟨ **generate**
output
update ⟩
test($PROGRAM_STATE == run)
input
interpret
update ⟩

Turn taking in IBiS1

Turn taking is regulated by the following principle:

- if **select** finds a move to perform, the system will generate a string and output it to the user.
- The TIS is then updated, and provided the PROGRAM_STATE variable is still set to run, the system reads input from the user, interprets it, and again updates the TIS.

This means that if **select** finds no move to perform, the turn will be handed over to the user.

IBiS1 Datatypes

As a preliminary to the presentation of the semantics and update strategies used by IBiS1, we will list the system-specific types used by the system.

- Types related to semantics
  - Question
    - WHQ
    - YNQ
    - ALTQ
  - Proposition
  - ShortAns
  - Ind
- Types related to plans and actions
  - Action
  - PlanConstr
- Miscellaneous types
  - Participant
  - ProgramState

Outline

Semantics in IBiS1

Atom types

- Pred_n, where $n = 0$ or $n = 1$: n-place predicates, e.g., dest-city, month
- Ind: Individual constants, e.g., paris, april
- Var: Variables, e.g., $x, y, \ldots, Q, P, \ldots$

Semantics in IBiS1

Sentences

- *Expr* : Sentence iff *Expr* : Proposition or *Expr* : Question or *Expr* : ShortAns
- *Expr* : Proposition if
  - *Expr* : Pred_0 or
  - *Expr* = $pred_1(arg)$, where *arg* : Ind and $pred_1$: Pred_1 or
  - *Expr* = $\neg P$, where P : Proposition or
  - *Expr* = fail(q), where q : Question

Semantics in IBiS1

- *Expr* : Question if *Expr* : YNQ or *Expr* : WHQ or *Expr* : ALTQ
  - $?P$: YNQ if P : Proposition
  - $?x.pred_1(x)$: WHQ if x : Var and $pred_1$: Pred_1
  - $\{ynq_1, \ldots, ynq_n\}$: ALTQ if ynq_i : YNQ for all i such that $1 \leq i \leq n$
  - *Expr* : ShortAns if
    - *Expr* = yes or
    - *Expr* = no or
    - *Expr* : Ind or
    - *Expr* = $\neg arg$ where *arg* : Ind

Propositions

- In a dialogue system operating in a domain of limited size, it is often not necessary to keep a full semantic representation of utterances.
- For example, a user utterance of "I want to go to Paris" could normally be represented semantically as e.g.
  - want(user, go-to(user, paris)) or
  - want(u, go-to(u,p)) & city(p) & name(p, paris) & user(u).
- We will be using a semantic representation with a coarser, domain-dependent level of granularity
  - The above example will be rendered as dest-city(paris)
- This reduced representation can be regarded as a reflection of the level of semantic granularity inherent in the underlying domain task.
  - As an example of the latter, in a travel agency domain there is no point in representing the fact that it is the user (or customer) rather than the system (or clerk) who is going to Paris; it is implicitly assumed that this is always the case.

Questions

- *y/n*-questions are propositions preceded by a question mark, e.g., ?dest-city(london) ("Do you want to go to London?")
- *wh*-questions are lambda-abstracts of propositions, with the lambda replaced by a question mark, e.g., ?x.dest-city(x) ("Where do you want to go?")
- alternative questions are sets of *y/n*-questions, e.g. {?dest-city(london), ?dest-city(paris)} ("Do you want to go to London or do you want to go to Paris?")

Short answers

- Ginzburg uses the term "short answers" for phrasal utterances in dialogue such as "paris" in the dialogue above
- These are standardly referred to as *elliptical* utterances.
- Ginzburg argues that (syntactic) ellipsis, as it appears in short answers, is best viewed as a semantic phenomenon with certain syntactic presuppositions.
- That is, the syntax provides conditions on what counts as a short answer but the processing of short answers is an issue for semantics.

Short answers and semantic underspecification

- In a dialogue system application, what we are really interested in is semantic underspecification *with regard to the domain/activity*
- An utterance is semantically underspecified iff it does not determine a unique and complete proposition in the given activity
- This means that whether an utterance is regarded as underspecified or not depends on what distinctions are interesting in a certain activity
  - For example, given the semantics that we propose for the travel agency domain, "to paris" is not semantically elliptical, since it determines the complete proposition dest-city(paris).
  - However, "to Paris" would be semantically underspecified in an activity where it could also be taken to mean e.g. "You should go to Paris".

Semantic sortal restrictions

- IBiS uses a rudimentary system of domain-dependent semantic sortal categories.
- For example, the travel agency domain includes the sorts city, means_of_transport, class, etc.
- All members of Ind are assigned a sort; for example, the individual constant paris has sort city and flight has sort means_of_transport.
- Sorts make it possible to distinguish non-meaningful propositions from meaningful ones.
- However, what is meaningful in one activity may not be meaningful in another, and vice versa.
- Therefore, the sortal system is implemented as a part of the domain knowledge.

Structure

- The property of a proposition P being sortally correct is implemented in IBiS1 as sort-restr(P).
- A proposition is sortally correct if its arguments fulfil the sortal constraints of the predicate.
- For example, the proposition dest_city(X) is sortally correct if the sort of X is city.

TDM-XML syntax for declaring a sort, defining the sort of a predicate argument and sort of individual (in ontology.xml):

```
<sort name="city"/>
<predicate name="dest_city" sort="city"/>
<individual name="paris" sort="city"/>
```

Relations between questions and answers

- Ginzburg defines two relations between questions and answers that are used in dialogue management protocols: resolves and about.
- We adapt Ginzburg's definitions of these relations for use in IBiS1.

The *resolves* relation

Question	Resolving answers
$?x.pred_1(x)$	a and $pred_1(a)$
$?P$	yes, no, P, and $\neg P$
$\{?P_1, ?P_2, \ldots, ?P_n\}$	P_i, $1 \leq i \leq n$

- General constraint: the sort of the answer must match the sortal constraint on the predicate in the question
- Example: the content of a resolving answer to a *wh*-question $?x.\text{dest-city}(x)$ about destination city is either
  - a proposition of the form $\text{dest-city}(C)$ or
  - an underspecified propositional content C
  - where C has the semantic sort city

Relevance (a.k.a. aboutness)

- All resolving answers are relevant:
  - relevant(A, Q) if resolves(A, Q), where A : Proposition or A : ShortAns, and Q : Question
- Relevant but not resolving answers to questions:

Question	Relevant, non-resolving answers
$?x.pred_1(x)$	$\neg a$, $\neg pred_1(a)$
$\{?P_1, ?P_2, \ldots, ?P_n\}$	$\neg P_i$, $1 \leq i \leq n$
Q	fail(Q)

Combining Questions and Answers to form Propositions

- A question q and an answer a *combine* to form a proposition p.
- General constraint: all resulting propositions must be sortally correct, i.e. the sort of the individual must match the sortal constraint on the predicate

Question	Answer	Proposition
$?x.pred_1(x)$	a or $pred_1(a)$ $\neg a$ or $\neg pred_1(a)$	$pred_1(a)$ $\neg pred_1(a)$
$?P$	yes or P no or $\neg P$	P $\neg P$
$\{?P_1, ?P_2, \ldots, ?P_n\}$	P_i, $(1 \leq i \leq n)$ $\neg P_i$, $(1 \leq i \leq n)$	P_i $\neg P_i$

Dialogue moves in IBiS1

- The following dialogue moves are used in IBiS1:
  - ask(q), where q : Question
  - answer(a), where a : ShortAns or a : Proposition
  - greet
  - quit
- In inquiry-oriented dialogue, the central dialogue moves concern raising (ask) and addressing (answer) issues
- The greet and quit moves are used in the beginning and end of dialogues to greet the user and indicate that the dialogue is over, respectively.

Interpreting utterances as moves

- Dialogue move types are often defined in terms of sentence mood, speaker intentions, and/or discourse relations
- Here, the type of move realized by an utterance is determined instead by the relation between the content of the utterance, and the activity in which the utterance occurs
  - an utterance is classified as realizing an answer-move only if its content is a relevant answer to some question available in the domain.
  - n utterance is classified as realizing an ask move only if its content is taken to be some question available (relevant) in the domain.
- The available questions are encoded in dialogue plans in the *domain knowledge* resource, either as issues to be resolved by a plan or as issues to be raised or resolved as part of a plan.
- The mapping from utterances to moves (move type plus content) is specified in the *lexicon* (TDM: *grammar*) resource.

Interpreting utterances as moves, cont'd

- On this approach, move classification does not rely on the dialogue context, nor (except to a very small extent) on syntactic form.
- Whether utterance u is interpreted as answer(A) is independent of whether a corresponding question has been raised.
- It is also independent of whether u is a declarative, interrogative, or imperative sentence, or a sentence fragment.
- Examples
  - "I want to go to Paris", "Can I go to Paris?", "Get me to Paris!" and "to Paris" are all interpreted as answer(dest-city(paris)).
  - "What's the price?", "Give me price information!", "I want to know about price.", "I want price information." and "price information" are all interpreted ask ask($?x$.price(x)).

Outline

Dialogue plans

- We introduce a simple formalism for representing procedural plans as sequences of actions.
- The plan representation syntax will be extended with additional plan constructs in later chapters to handle more complex plans, including branching conditions and subplans.
- The plan constructs in IBiS1 are either simple actions or action sequences.
- Action sequences have the form $\langle a_1, \ldots, a_n \rangle$ where a_i : Action $(1 \leq i \leq n)$.

Dialogue plans, cont'd

- Kinds of actions
  - All dialogue moves are actions.
  - There are also more abstract actions which are indirectly connected to dialogue moves.
  - Finally, there are actions which are not connected to specific dialogue moves.
- In the following, q is a question, and p is a proposition.
- (To avoid confusion, one may want to distinguish *dialogue actions*, which are part of dialogue plans, from *domain actions* which are the kinds of things one can do in a certain domain, e.g. call someone)

Actions connected to dialogue moves

- findout(q): find the answer to q.
  - In IBiS1, this is done by asking a question to the user, i.e., by performing an ask move.
  - The findout action is not removed until the question q has been resolved.
- raise(q): raise the question q.
  - This action is similar to findout, except that it is removed from the plan as soon as the ask-move is selected.
  - This means that if the user does not answer the question when it has been raised, it will not be raised again.
  - This is useful e.g., for requesting optional information.
- respond(q): provide an answer to question q.
  - This is done by performing an answer move with content p, where p is a resolving answer to q.

Accessing external device

- The database consultation action is not connected to any specific dialogue move.
- Database consultation can be seen as a special case of *domain actions*.
- The domain actions are determined by the application.
- The set of available domain actions are defined as relations or updates in the resource interface for the application.
- For a typical information-exchange task, the application is a database
- (In TDM, as well as in the action-oriented version of IBiS, there are several additional plan constructs for accessing external devices)
- For our database application in the travel agency domain, we use a single domain action consultDB(q) (where q is a question)

A sample IBiS1 plan

ISSUE : $?x.\text{price}(x)$

PLAN: $\langle$

findout($?x.\text{means\_of\_transport}(x)$),

findout($?x.\text{dest\_city}(x)$),

findout($?x.\text{depart-city}(x)$),

findout($?x.\text{depart-month}(x)$),

findout($?x.\text{depart-day}(x)$),

raise($?x.\text{class}(x)$),

consultDB($?x.\text{price}(x)$)

$\rangle$

A sample TDM plan

```
<goal type="resolve" question_type="wh_question"
      predicate="price">
  <plan>
    <findout type="wh_question" predicate="means_of_transport"/>
    <findout type="wh_question" predicate="dest_city"/>
    <findout type="wh_question" predicate="depart_city"/>
    <findout type="wh_question" predicate="depart_month"/>
    <findout type="wh_question" predicate="depart_day"/>
    <raise type="wh_question" predicate="class"/>
    <dev_query device="TravelDatabaseDevice" type="wh_question"
               predicate="price"/>
  </plan>
</goal>
```