Type-based Human-Computer Dialogue

Peter Ljunglöf

Dept. of Computer Science and Engineering
University of Gothenburg / Chalmers Univ. of Technology

The TYPES meeting
Toulouse, 26 April 2013

Dialogue systems

A dialogue system interacts with humans via natural language

- spoken dialogue (telephone, in-car)
- written dialogue (web helper agents, chatbots)
- multimodal dialogue (virtual worlds, avatars, information kiosks)

The interaction should be longer than a simple Q/A pair

- i.e., IBM Watson or Apple Siri are *not* dialogue systems
- we need to be able to keep track of the state in the dialogue
- we need a way to specify the dialogue domain and structure

Different kinds of dialogue systems

- Finite-state: the dialogue is defined as a finite-state flowchart
  - \+ simple implementation
  - – each possible interaction sequence has to be implemented
- Form-based: the dialogue consists of filling slots in a form
  - \+ more freedom in how to fill the slots
  - \+ the most common framework (VoiceXML is a W3C standard)
  - – if you have several forms, you first have to select the form
  - – limited possibilities in creating hierarchical forms
- Information-state update rules with preconditions / effects (ISU)
  - \+ allows for sub-dialogues, incremental dialogue, etc.
  - \+ multiple interaction modes (system-, user-driven, mixed initiative)
  - – update rules can affect other rules (butterfly effect)
  - – no underlying mathematical theory

The information-state update approach (ISU)

An ISU dialogue system consists of an information state (IS)

- the IS models the beliefs, desires and intentions (BDI) of the dialogue participants

The IS is modified by update rules

- rules are defined by preconditions and effects on the IS

The dialogue utterances are encoded as dialogue moves (DM)

- examples: $ask(\ldots)$, $answer(\ldots)$, $request(\ldots)$, $feedback(\ldots)$
- the IS contains specific slots for incoming and outgoing DMs

Converting between utterances and DMs is a problem of its own

- but it can be done in type theory (e.g., Luo, Retoré, Ranta, Cooper)

Using type theory for dialogue

In this talk I present an idea of how to use type theory for dialogue

- the domain is specified by a list of defined constants and functions
  - i.e., what the dialogue system can talk about
- the goal of a dialogue is to build a complete, type-correct term
  - during the dialogue, the term is partial and contains metavariables
  - the term is a part of the information state,
    and represents the beliefs and intentions of the participants
- the term is built interactively, much as an interactive proof editor
  - initially, the term is a single metavariable of the goal type, ?*Goal*
  - the system asks questions corresponding to the term's metavariables
  - the user gives answers that refine the metavariables

An example domain

A specification of a travel agency

$book : Event \to Goal$
$price : Event \to Goal$
$info : City \to Goal$

$hotelvisit : City \times Date \to Event$
$oneway : Route \times Date \to Event$
$return : Route \times Date \times Date \to Event$

$route : City \times City \times Means \to Route$

$today, tomorrow : Date$
$date : Month \times Day \to Date$

$tou, gbg, \ldots : City$

$flight, boat, \ldots : Means$

$jan, feb, \ldots : Month$

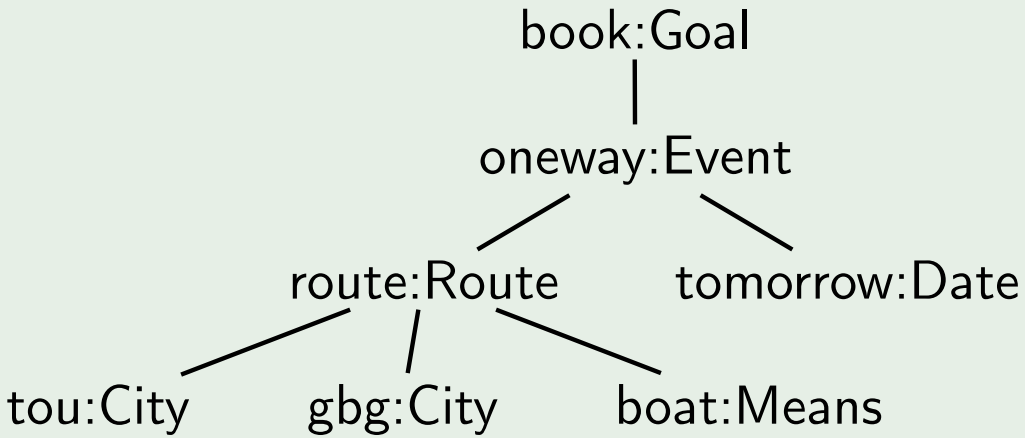
$1st, 2nd, \ldots : Day$

Terms and trees

A term that specifies a booking of a boat trip to Toulouse

$$book(oneway(route(tou, gbg, boat), tomorrow)) : Goal$$

The corresponding tree, with type annotation



Dialogue management by successive refinement

The dialogue system builds a complete term by successive refinement:

- the system asks questions corresponding to metavariables
- the user gives answers (but not necessarily to asked questions. . .)

The term can contain several metavariables, and one of them is active:

- it is called the focus node (and is shown highlighted)
- the initial term is the single focused node $?Goal$

When the user has given an answer to a metavariable,
the system moves the focus to another node

- when the term is complete, the dialogue is finished
- e.g., then the system can perform the requested booking

Two kinds of system questions

When it is the system's turn, it asks a question:

- if necessary, it moves the focus to a new metavariable
- it asks the question corresponding to the focus node

There are two possible questions for a given metavariable:

- a wh-question is written $?T$
  - $?Goal \Rightarrow$ "What do you want to do?"
- an alternative question is written $?f_1 \vee \ldots \vee f_n : T$
  - $?book \vee info : Goal$
    $\Rightarrow$ "Do you want to book an event or get city information?"

I won't discuss how to perform translation between utterances and terms (parsing and generation), but one possibility is to use type theory...

Example: System-driven dialogue

?Goal

S: ask(?*Goal*) $\Rightarrow$
"What do you want to do?"

Example: System-driven dialogue

?Goal

U: (. . . no answer. . .)

Example: System-driven dialogue

?book∨price∨info:Goal

S: ask($?book \vee price \vee info : Goal$) $\Rightarrow$
"Do you want to book an event, know the price or get city information?"

Example: System-driven dialogue

U: "Book an event" $\Rightarrow$
answer($book(?Event) : Goal$)

Example: System-driven dialogue

S: ask(?*Event*) $\Rightarrow$
"Which event are you interested in?"

Example: System-driven dialogue

U: (. . . incomprehensible. . .)

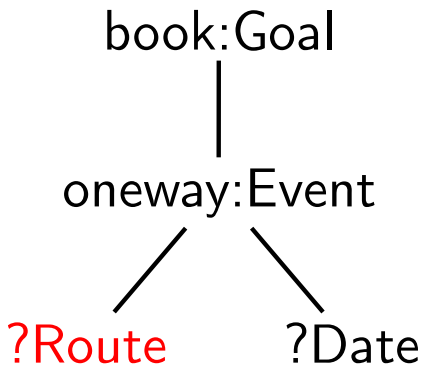
Example: System-driven dialogue

S: ask(?*hotelvisit* ∨ *oneway* ∨ *return* : *Event*) ⇒
"Do you want a hotel visit, a oneway trip or a return trip?"

Example: System-driven dialogue

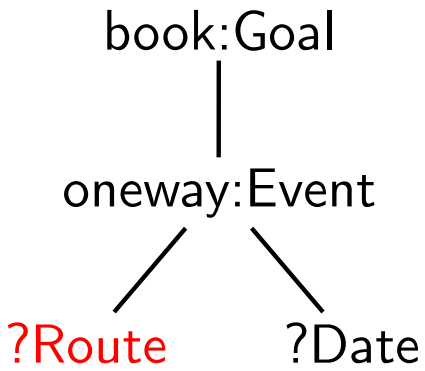
U: "A oneway trip" $\Rightarrow$
answer($oneway(?Route, ?Date) : Event$)

Example: System-driven dialogue



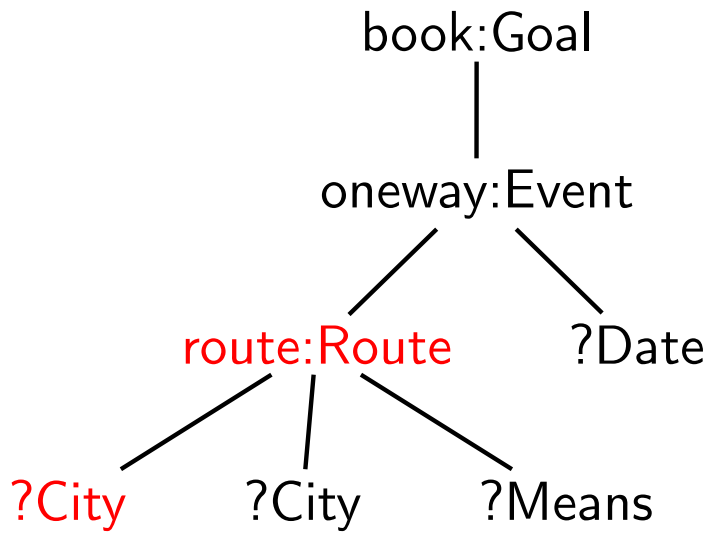
S: ask(?*Route*) $\Rightarrow$
"Which route do you want?"

Example: System-driven dialogue



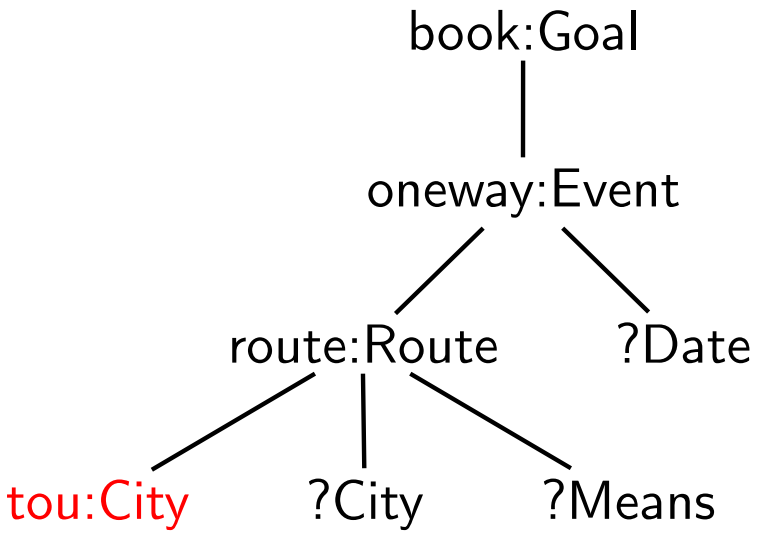
U: "Eh. . . ?"

Example: System-driven dialogue



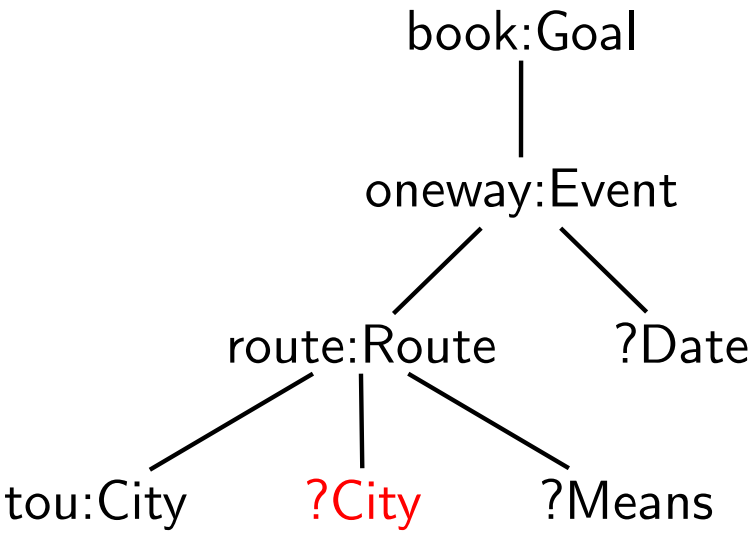
S: ask(?*City*) $\Rightarrow$
"To which city are you heading?"

Example: System-driven dialogue



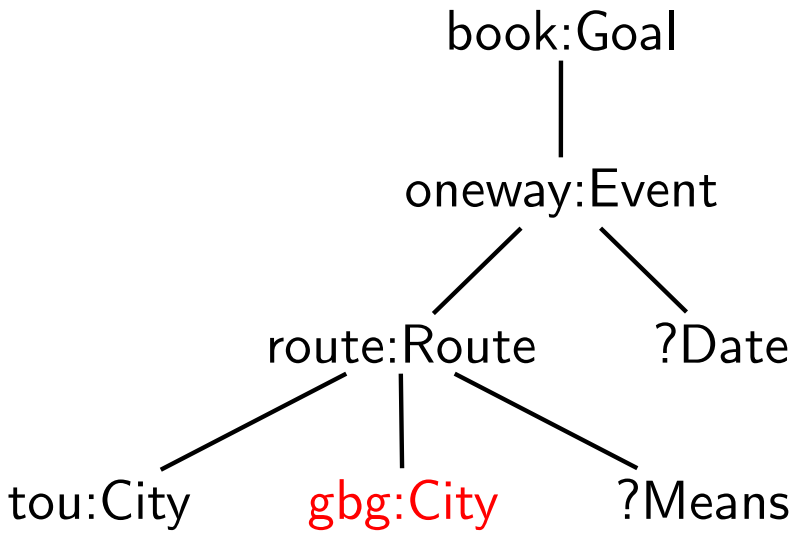
U: "To Toulouse" $\Rightarrow$
answer($tou : City$)

Example: System-driven dialogue



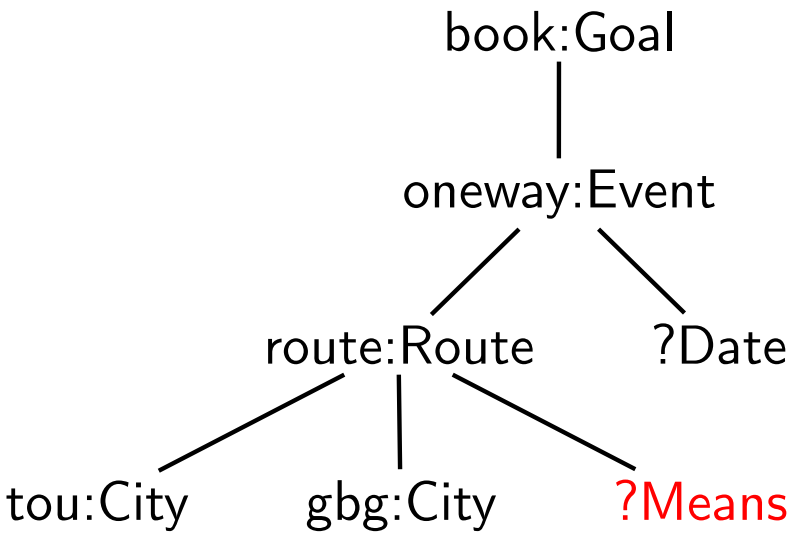
S: ask(?*City*) $\Rightarrow$
"From where are you leaving?"

Example: System-driven dialogue



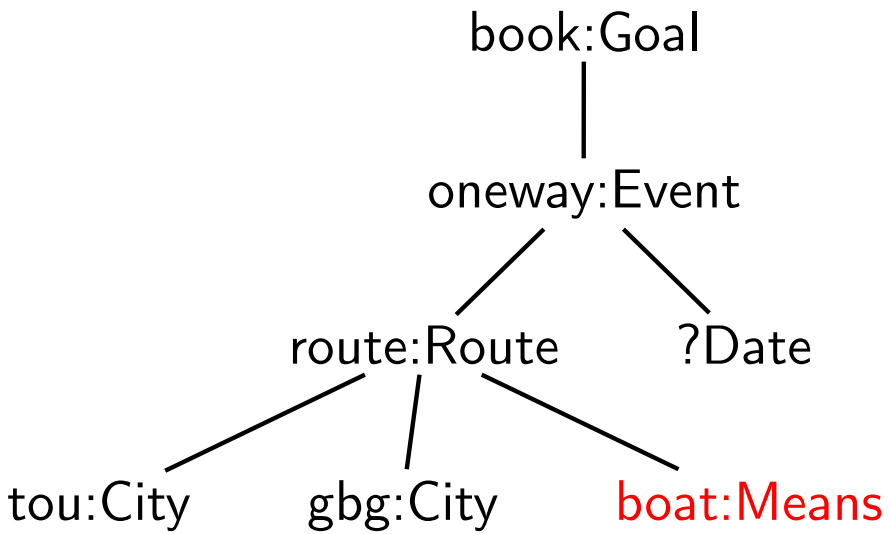
U: "From Gothenburg" $\Rightarrow$
answer($gbg : City$)

Example: System-driven dialogue



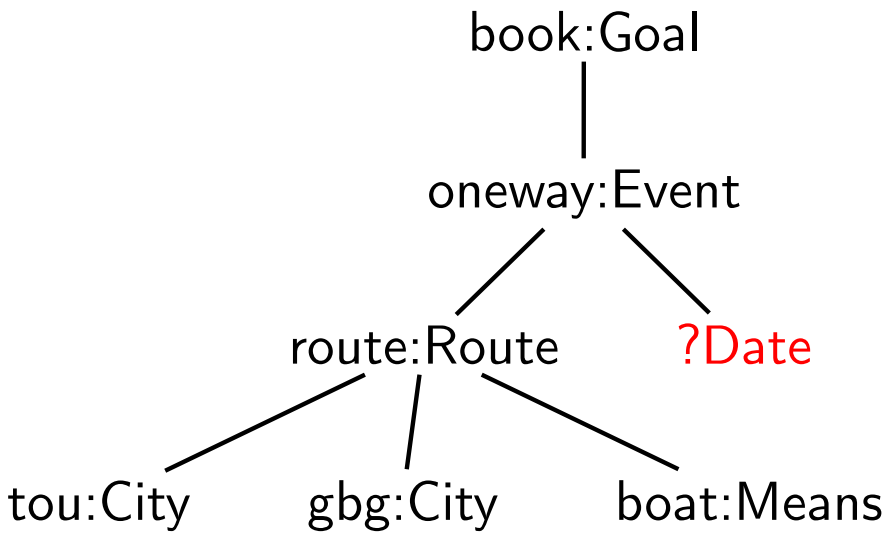
(etcetera. . .)

Example: System-driven dialogue



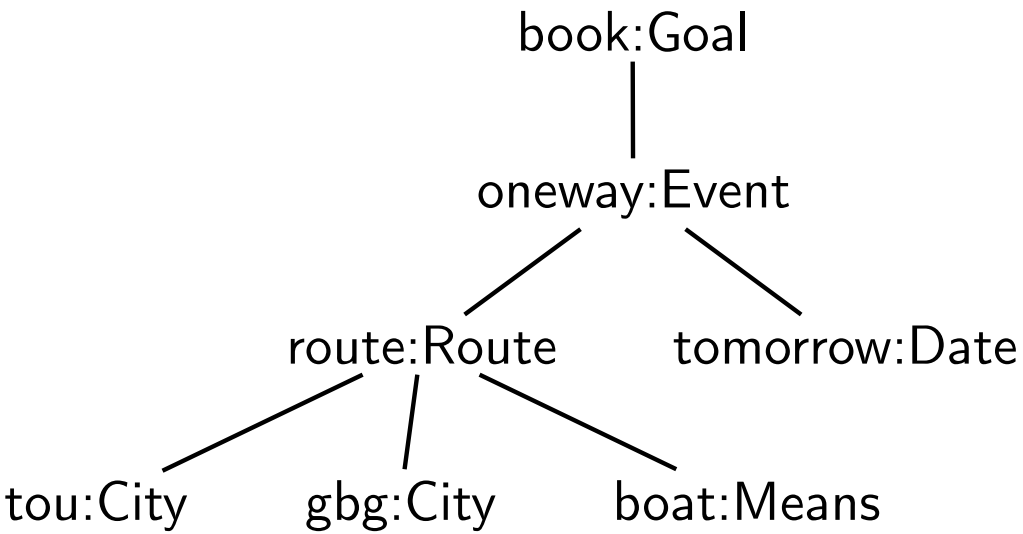
(etcetera. . .)

Example: System-driven dialogue



(etcetera. . .)

Example: System-driven dialogue



(etcetera. . .)

All this is old stuff – and inflexible too!

This dialogue system was inspired from the Alfa proof editor

- and was described by Ranta and Cooper (2004)

The dialogue works fine

- provided that the user always answers the system's questions...
- ***...which humans almost never do***

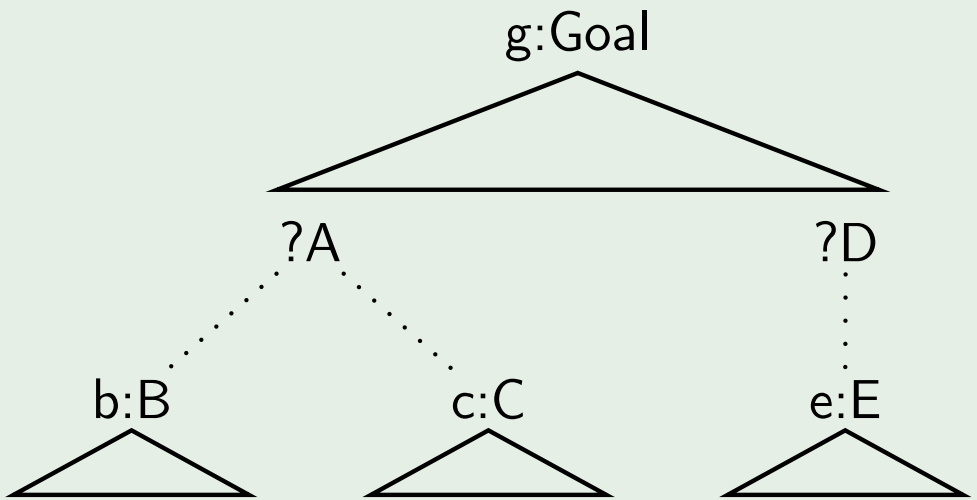
The system cannot handle underspecified answers

- such as "a boat trip to Toulouse" as the first answer
- which is ***not*** a term of type *Goal*, but instead of type *Route*
- ***...which is what humans almost always do***

I will now present my ideas on how underspecification can be handled

Underspecified tree nodes

I use underspecified tree nodes for incorporating underspecified information: when the user says something which the system cannot integrate into the current tree.



- the type A (resp. D) must dominate B, C (resp. E)
- the intuition is similar to proving a lemma before you know exactly where it fits in the final proof

Refinement strategies

When the user gives an underspecified answer to a question, the system has to figure out what to do next

- since the answer is underspecified, the current focus node will still be a metavariable
- but to continue asking the same question over and over again will seem very uncooperative

So the system can instead change its focus to show to the user that it has accommodated the answer

- this can be done in (at least) three different ways
- I call them top-down, bottom-up and "bottom-down"

Strategy 1: Top-down refinement

There are (at least) the following refinement strategies:

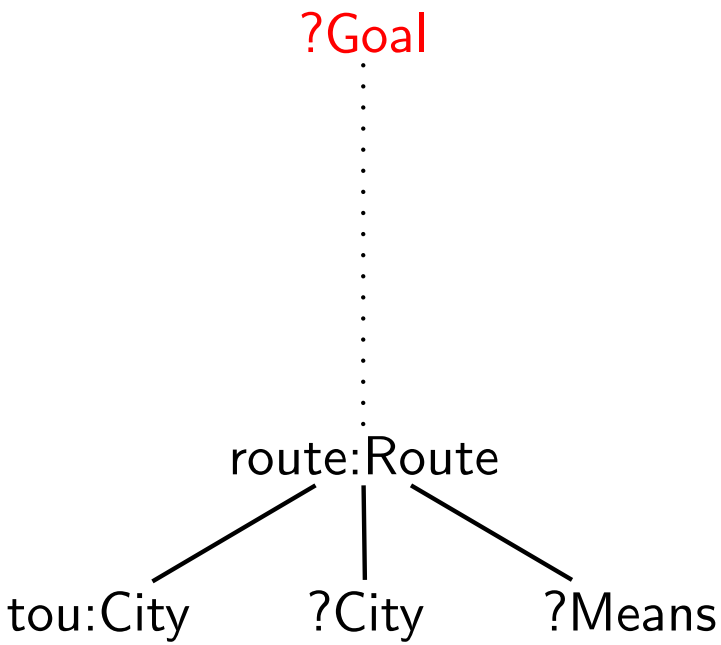
1. top-down refinement
2. bottom-up refinement
3. "bottom-down" refinement

Strategy 1: Top-down refinement

?Goal

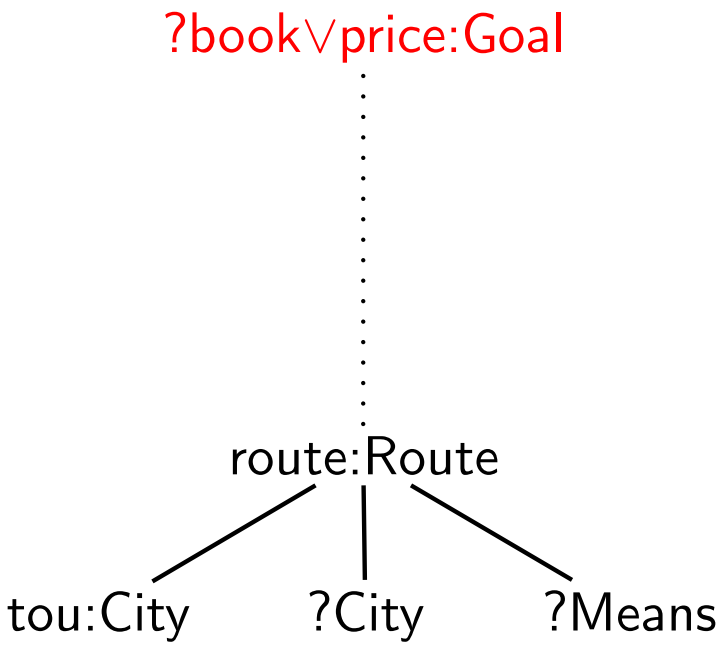
S: ask(?*Goal*) $\Rightarrow$
"What do you want to do?"

Strategy 1: Top-down refinement



U: "Go to Toulouse" $\Rightarrow$
answer($route(tou, ?City, ?Means) : Route$)

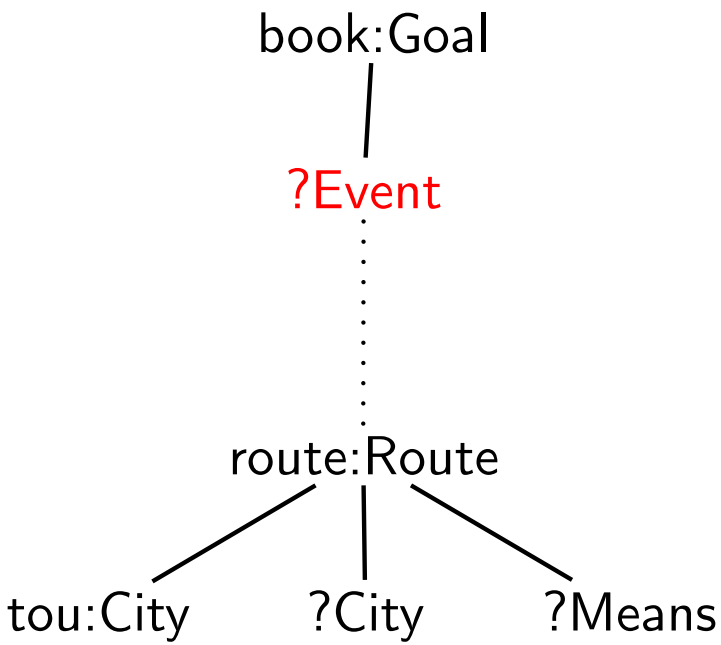
Strategy 1: Top-down refinement



The question is filtered by the underspecified node!

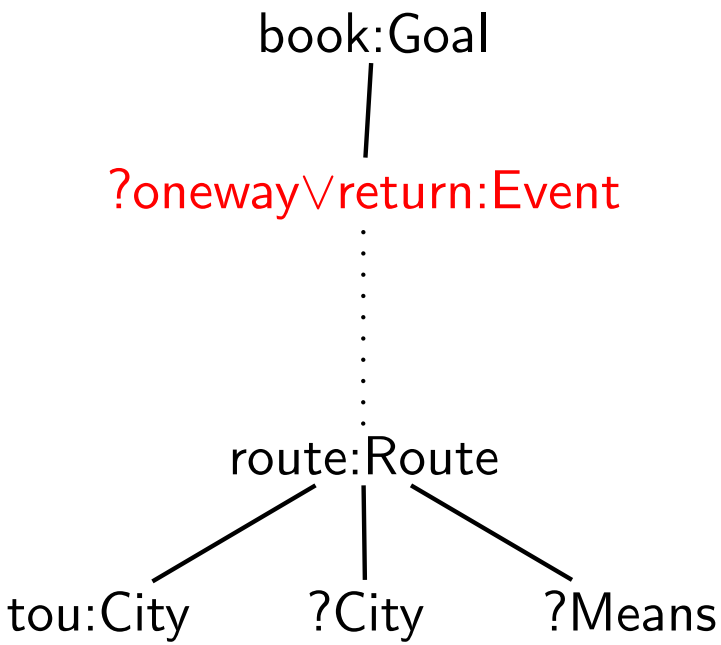
S: ask(?*book* ∨ *price* : *Goal*) ⇒
"Do you want to book an event or know the price?"

Strategy 1: Top-down refinement



U: "Book an event" $\Rightarrow$
answer($book(?Event) : Goal$)

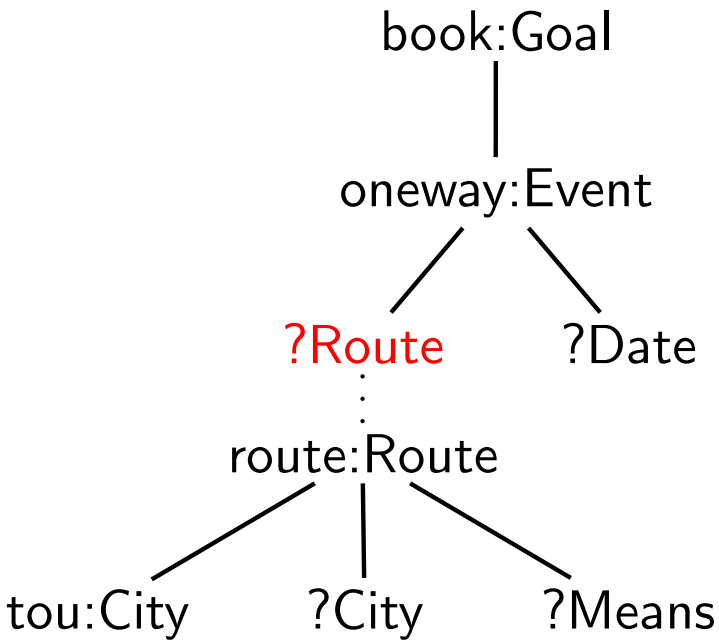
Strategy 1: Top-down refinement



The question is filtered by the underspecified node!

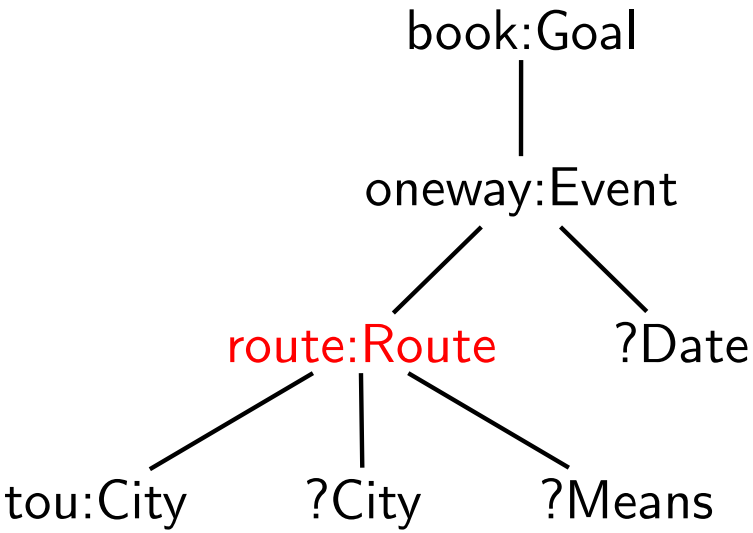
S: ask(?*oneway* ∨ *return* : *Event*) ⇒
"Do you want a oneway trip or a return trip?"

Strategy 1: Top-down refinement

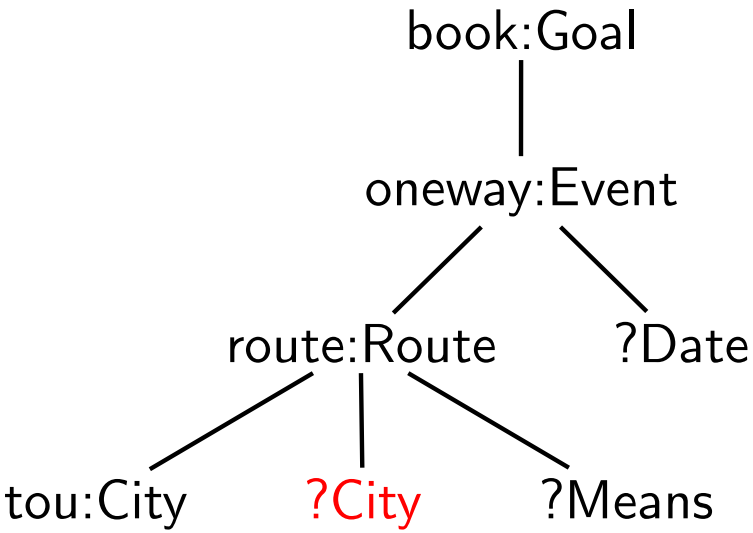


U: "A oneway trip" $\Rightarrow$
answer($oneway(?Route, ?Date) : Event$)

Strategy 1: Top-down refinement

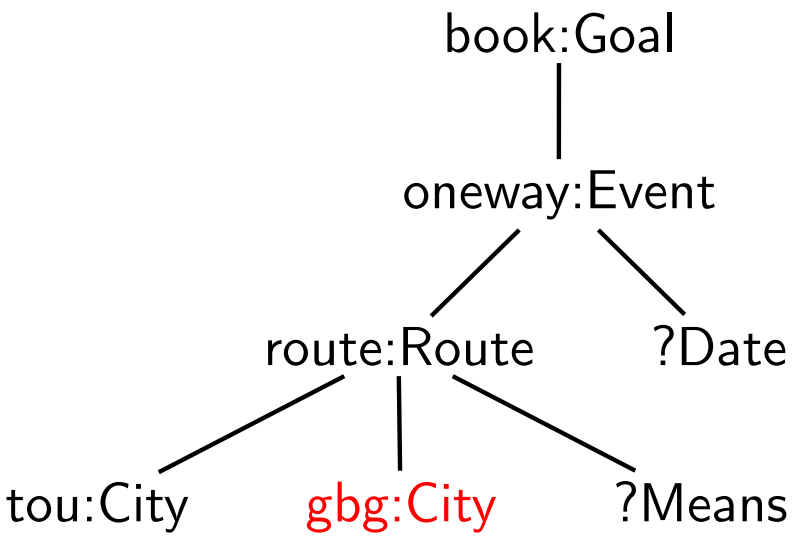


Strategy 1: Top-down refinement



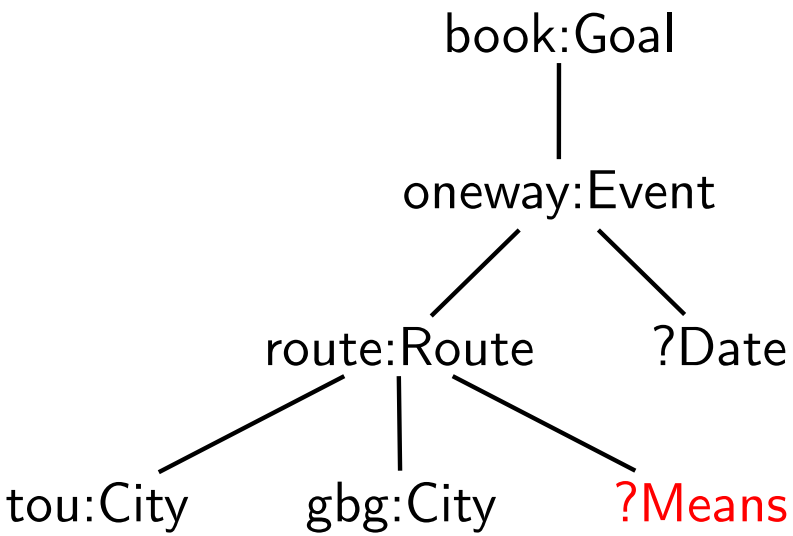
S: ask(?*City*) $\Rightarrow$
"From where are you leaving?"

Strategy 1: Top-down refinement



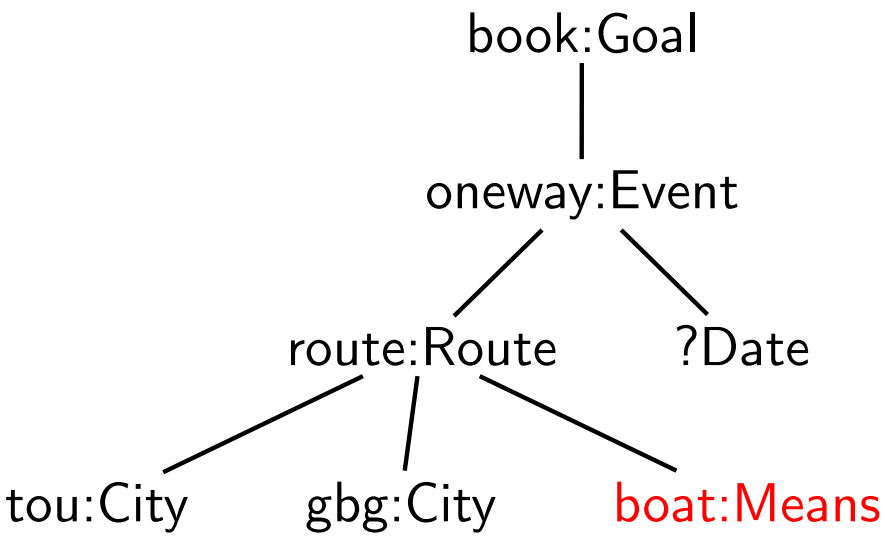
U: "Gothenburg" $\Rightarrow$
answer($gbg : City$)

Strategy 1: Top-down refinement



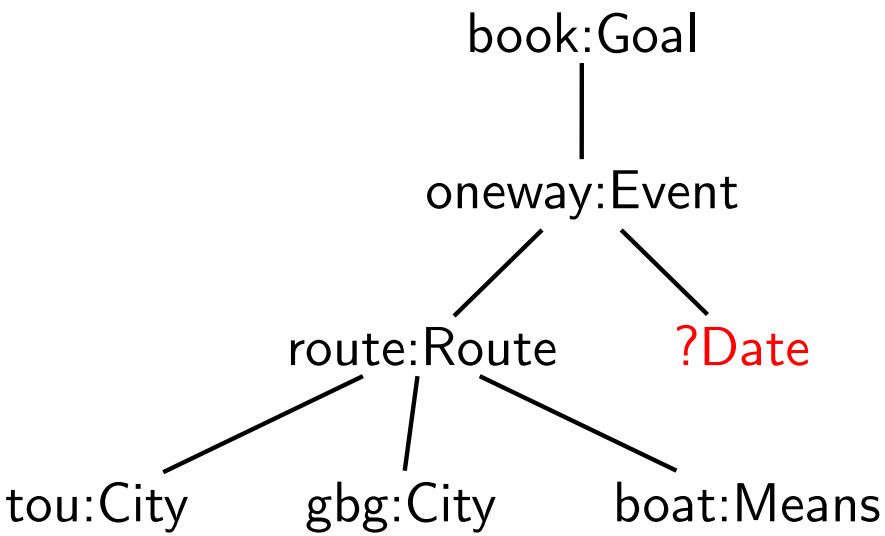
(etcetera. . .)

Strategy 1: Top-down refinement



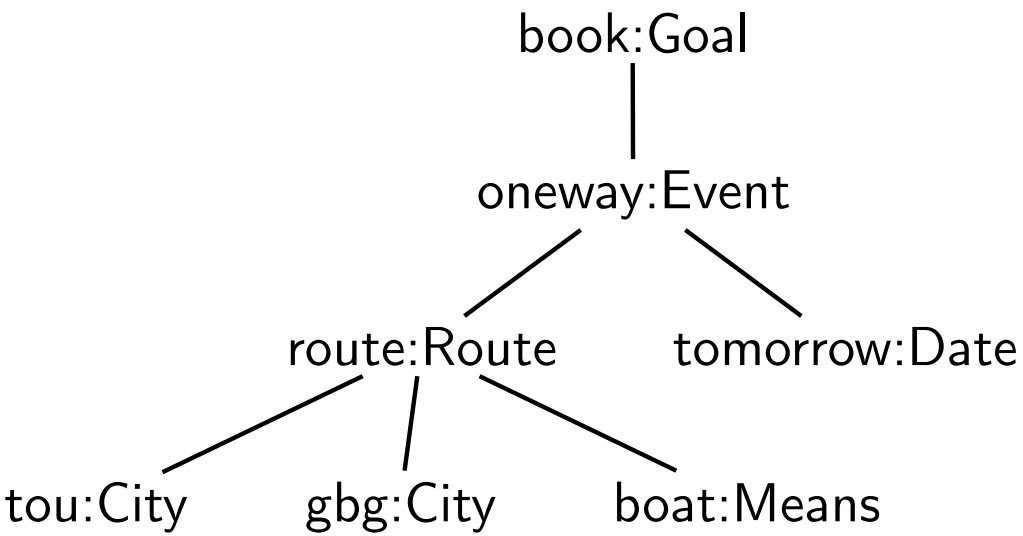
(etcetera. . .)

Strategy 1: Top-down refinement



(etcetera. . .)

Strategy 1: Top-down refinement



(etcetera. . .)

Strategy 2: Bottom-up refinement

There are (at least) the following refinement strategies:

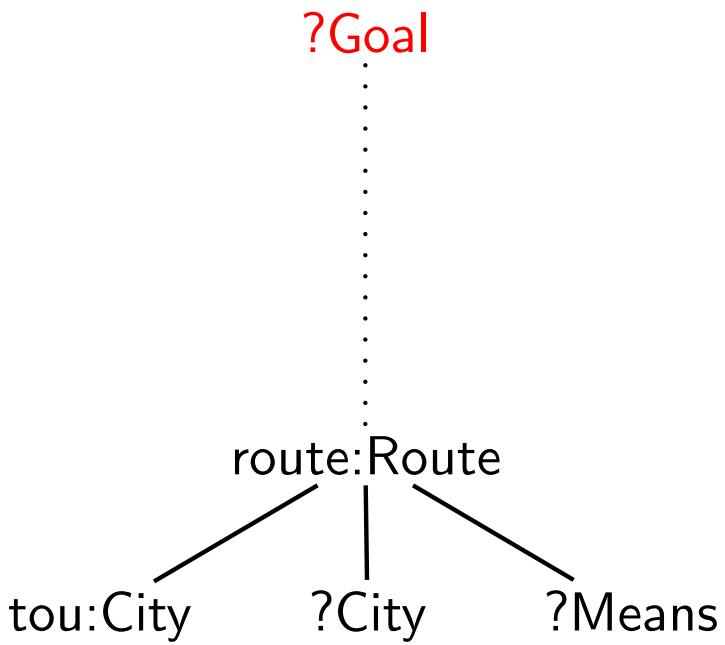
1. top-down refinement
2. bottom-up refinement
3. "bottom-down" refinement

Strategy 2: Bottom-up refinement

?Goal

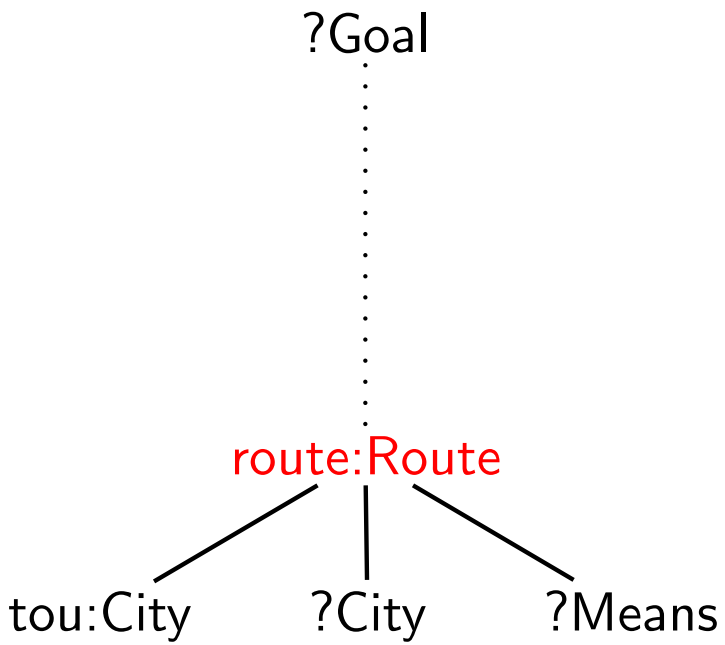
S: ask(?*Goal*) $\Rightarrow$
"What do you want to do?"

Strategy 2: Bottom-up refinement

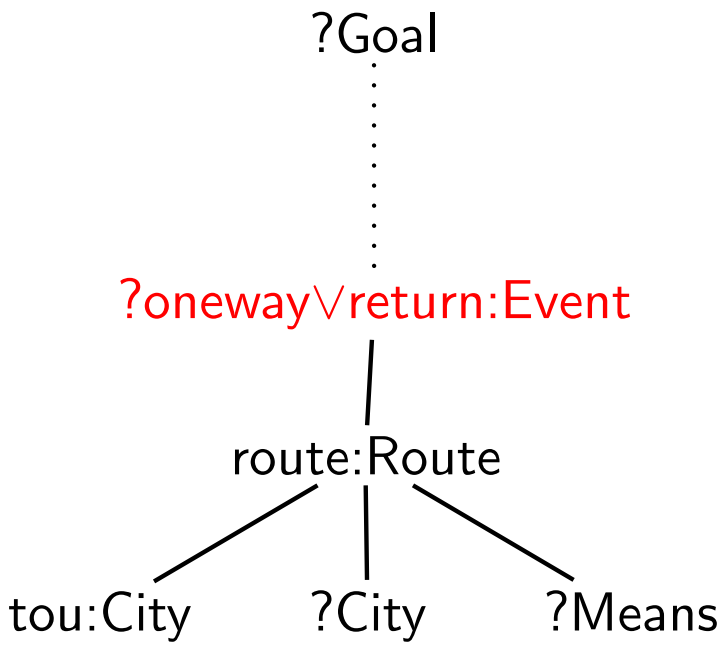


U: "Go to Toulouse" $\Rightarrow$
answer($route(tou, ?City, ?Means) : Route$)

Strategy 2: Bottom-up refinement



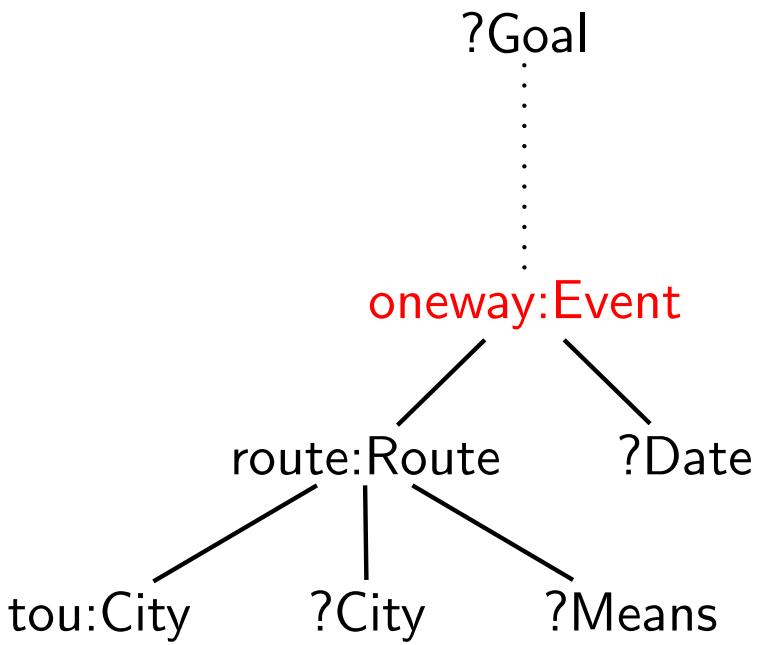
Strategy 2: Bottom-up refinement



The question is filtered by the goal type!

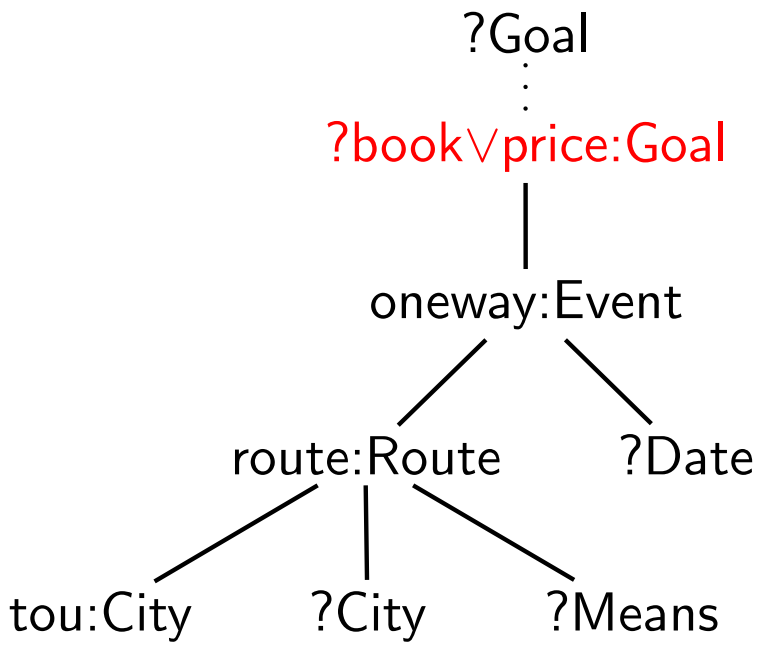
S: ask(?*oneway* ∨ *return* : *Event*) ⇒
"Do you want a oneway trip or a return trip?"

Strategy 2: Bottom-up refinement



U: "A oneway trip" $\Rightarrow$
answer($oneway(?Route, ?Date) : Event$)

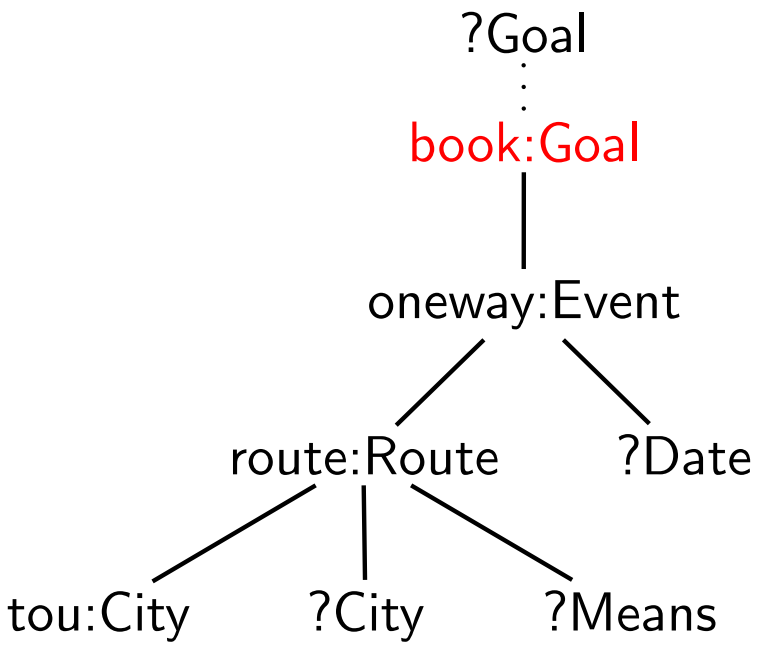
Strategy 2: Bottom-up refinement



The question is filtered by the goal type!

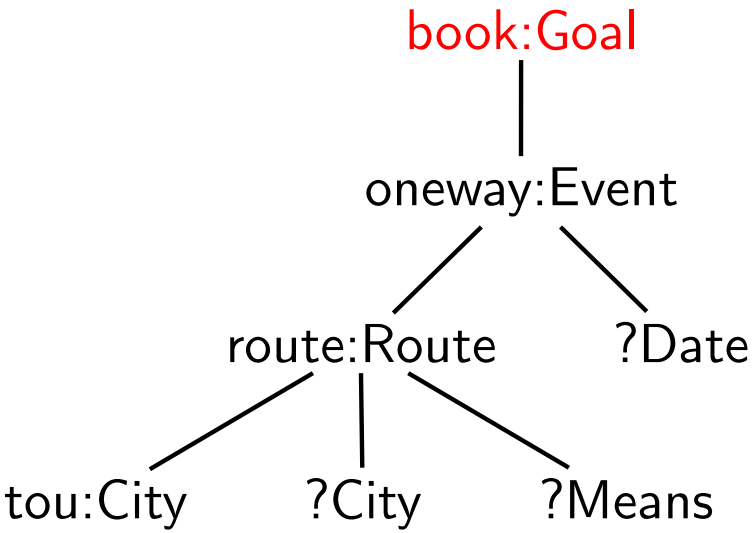
S: ask(?*book* ∨ *price* : *Goal*) ⇒
"Do you want to book an event or know the price?"

Strategy 2: Bottom-up refinement

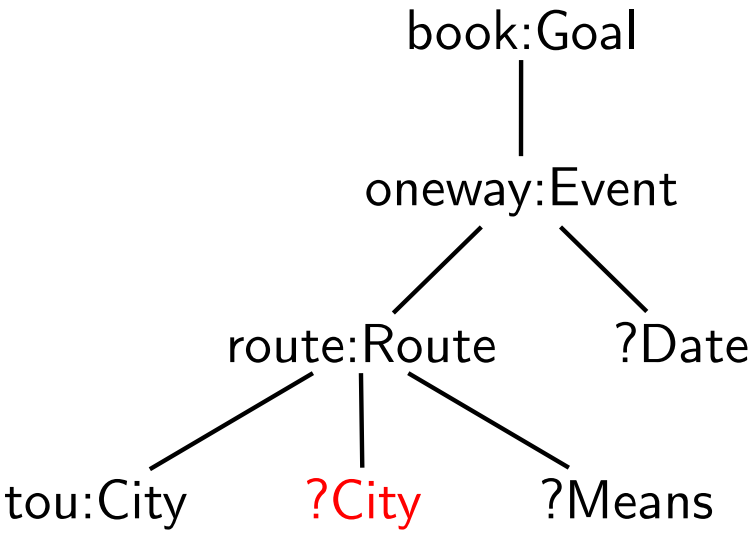


U: "Book an event" $\Rightarrow$
answer($book(?Event) : Goal$)

Strategy 2: Bottom-up refinement

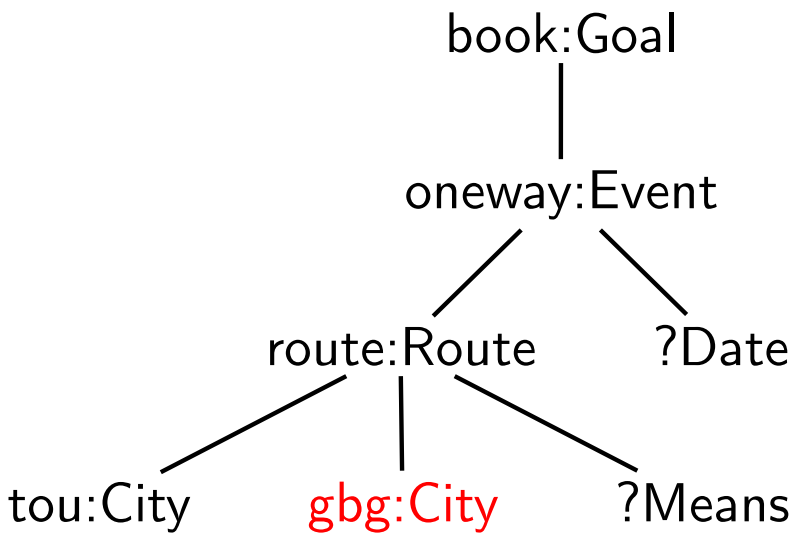


Strategy 2: Bottom-up refinement



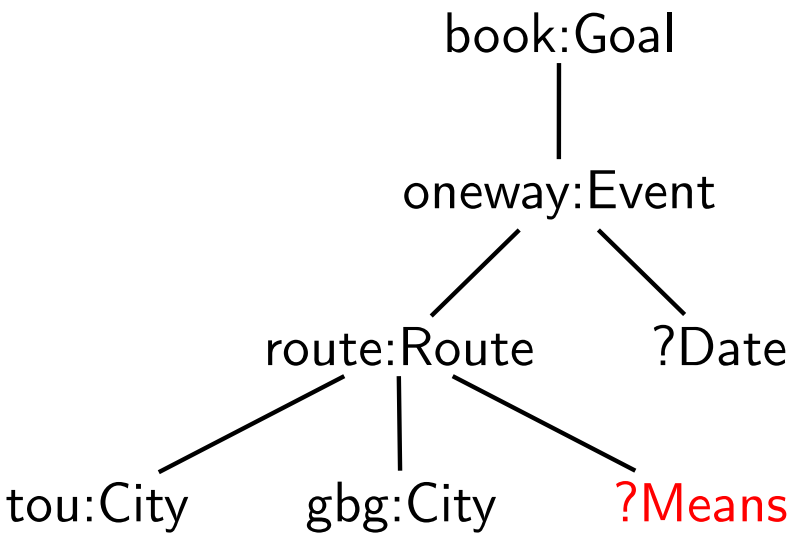
S: ask(?*City*) $\Rightarrow$
"From where are you leaving?"

Strategy 2: Bottom-up refinement



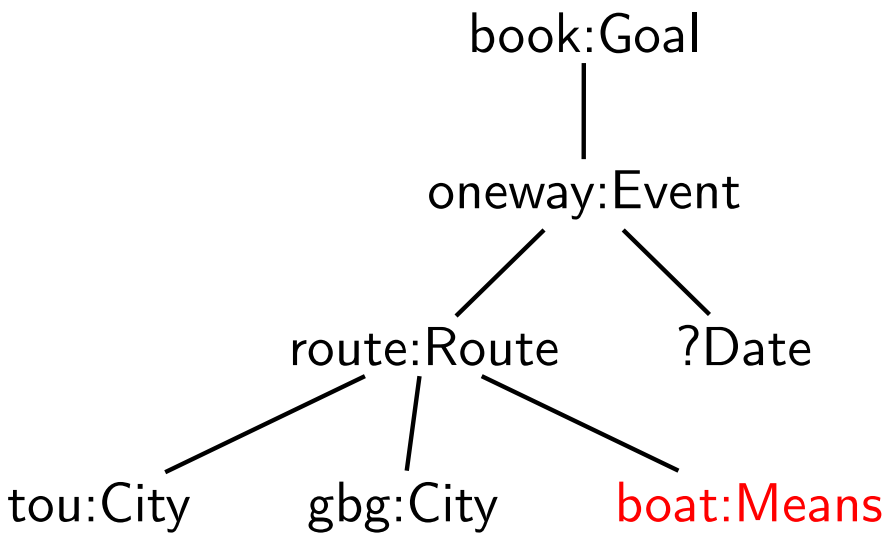
U: "Gothenburg" $\Rightarrow$
answer($gbg : City$)

Strategy 2: Bottom-up refinement



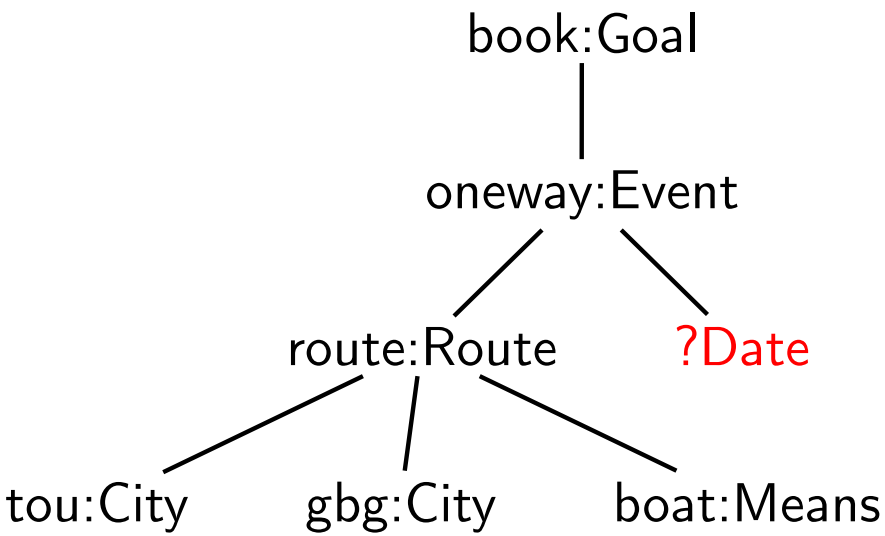
(etcetera. . .)

Strategy 2: Bottom-up refinement



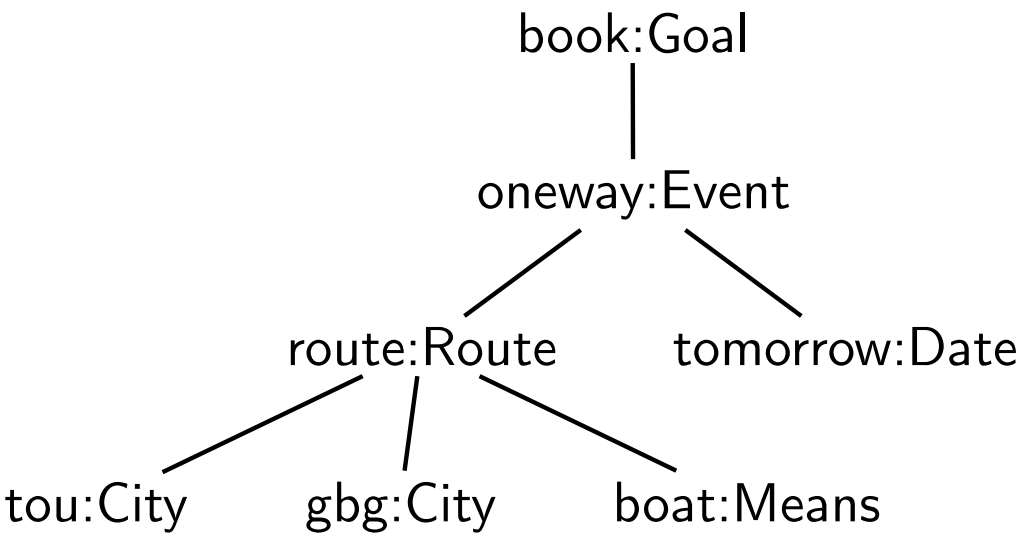
(etcetera. . .)

Strategy 2: Bottom-up refinement



(etcetera. . .)

Strategy 2: Bottom-up refinement



(etcetera. . .)

Strategy 3: "Bottom-down" refinement

There are (at least) the following refinement strategies:

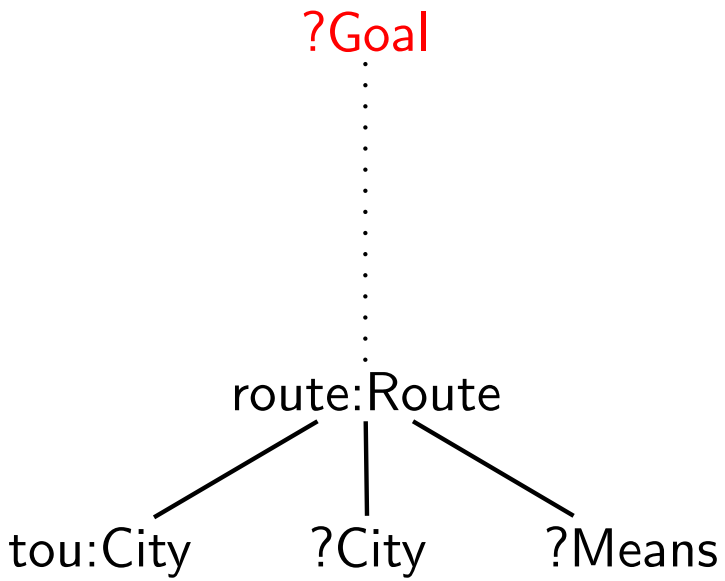
1. top-down refinement
2. bottom-up refinement
3. "bottom-down" refinement

Strategy 3: "Bottom-down" refinement

?Goal

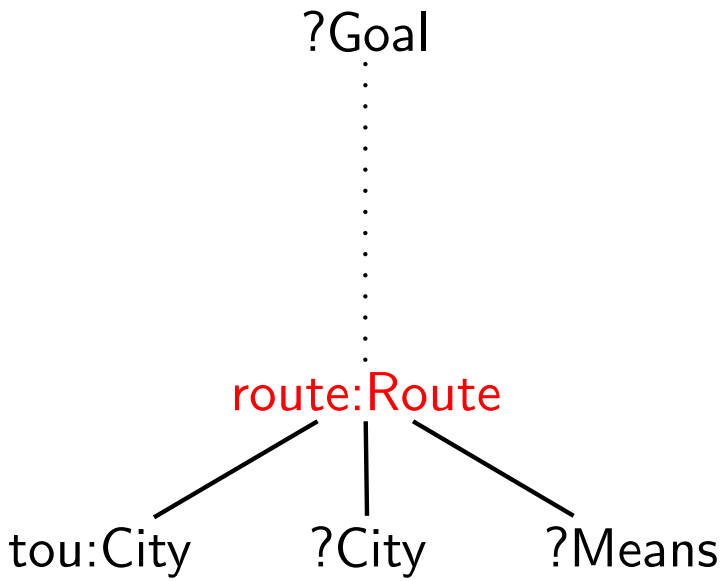
S: "What do you want to do?"

Strategy 3: "Bottom-down" refinement

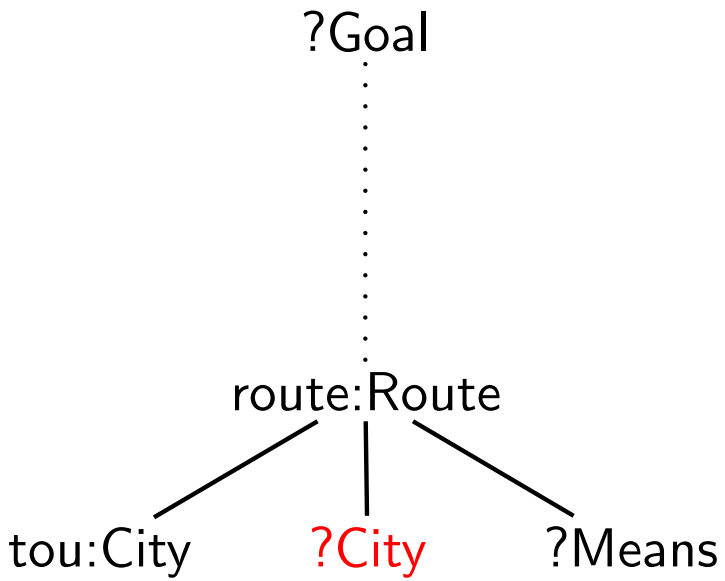


U: "Go to Toulouse"

Strategy 3: "Bottom-down" refinement

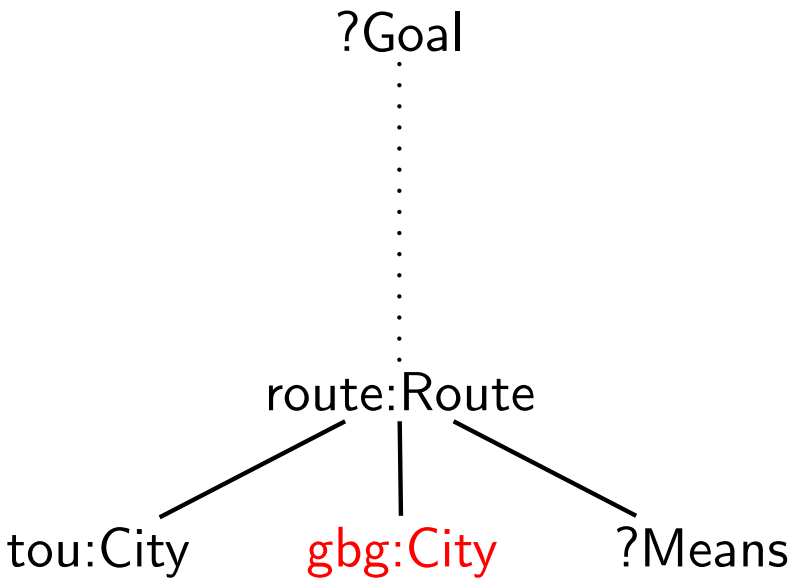


Strategy 3: "Bottom-down" refinement



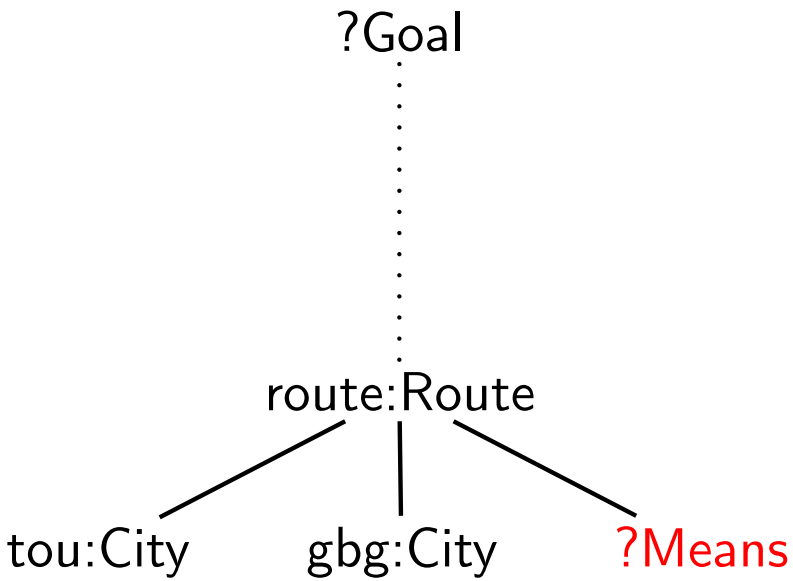
S: "From where do you want to go?"

Strategy 3: "Bottom-down" refinement



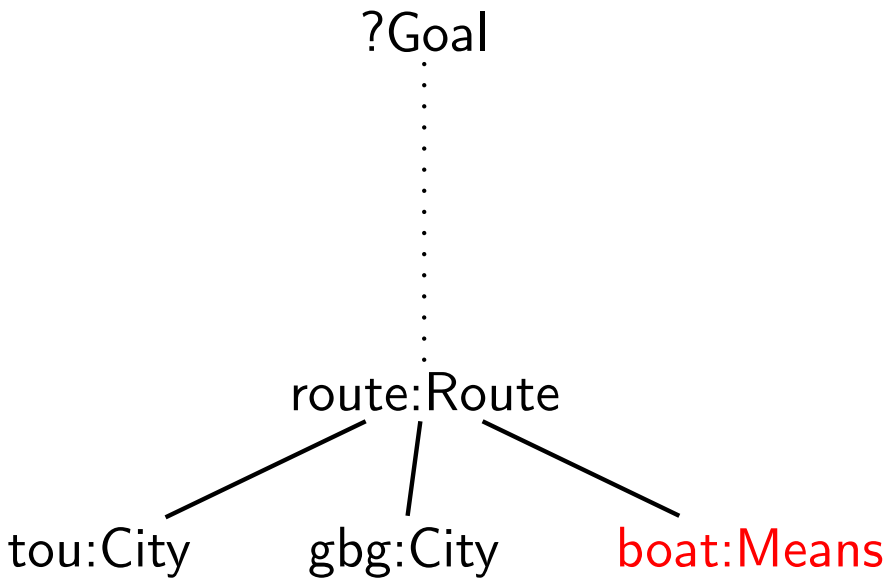
U: "From Gothenburg"

Strategy 3: "Bottom-down" refinement



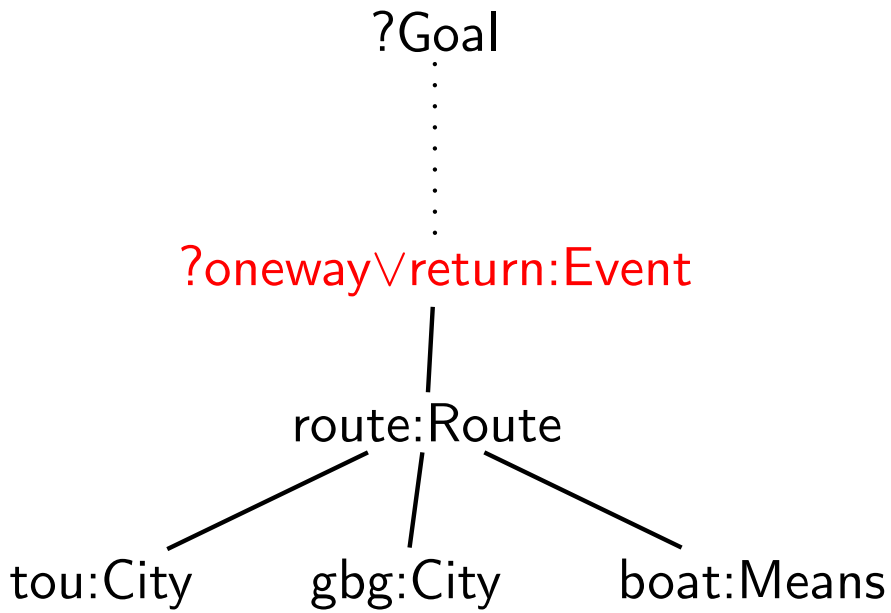
S: "How do you want to travel?"

Strategy 3: "Bottom-down" refinement



U: "By boat"

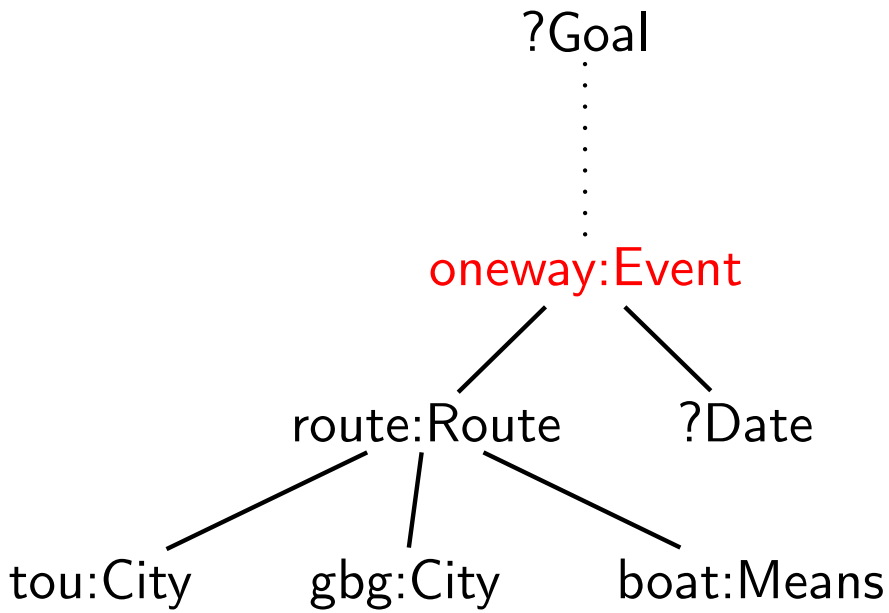
Strategy 3: "Bottom-down" refinement



The question is filtered by the goal type!

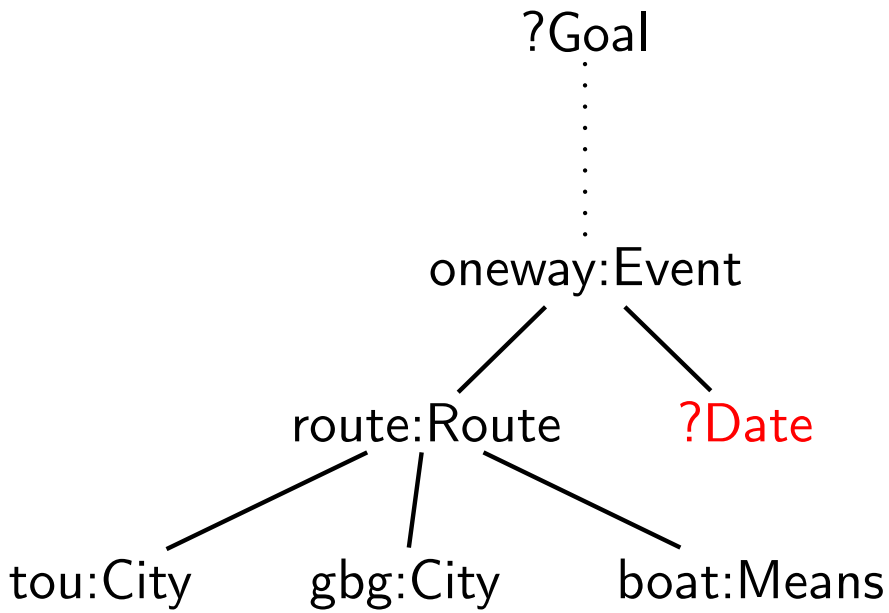
S: "Do you want a oneway or return trip?"

Strategy 3: "Bottom-down" refinement



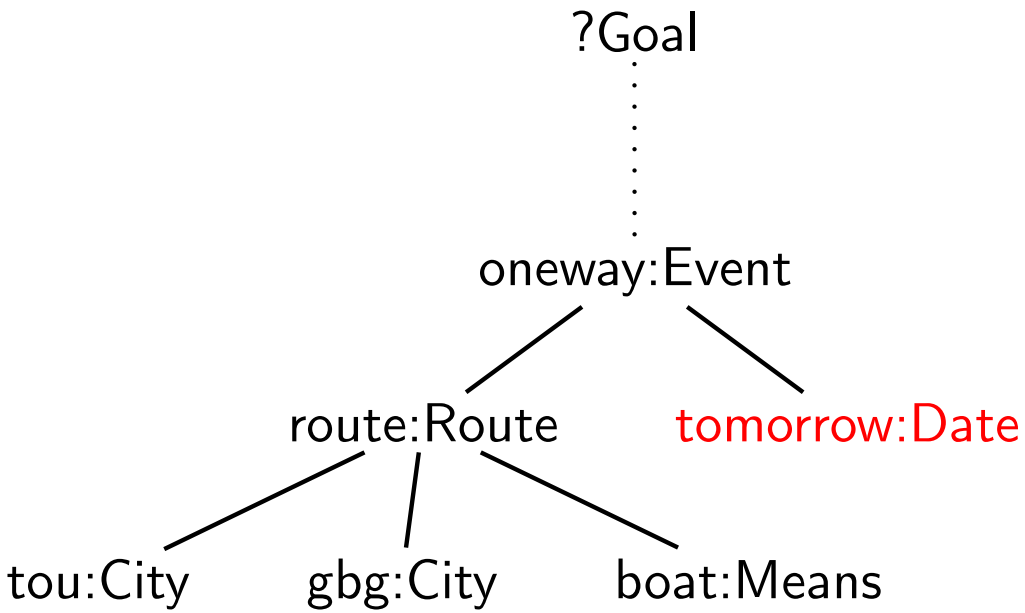
U: "Oneway"

Strategy 3: "Bottom-down" refinement



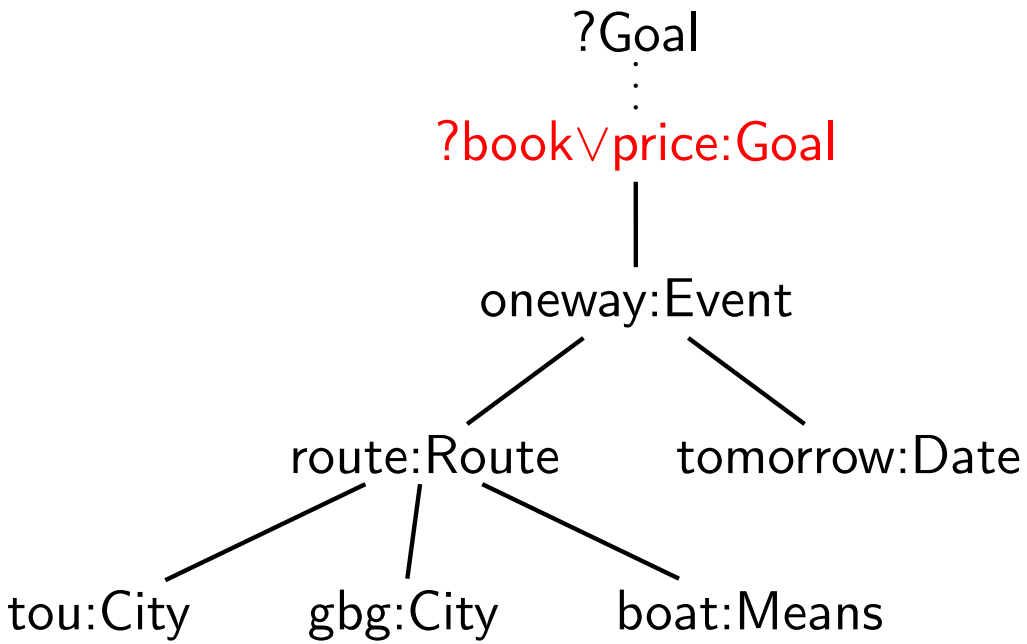
S: "When do you want to leave?"

Strategy 3: "Bottom-down" refinement



U: "Tomorrow"

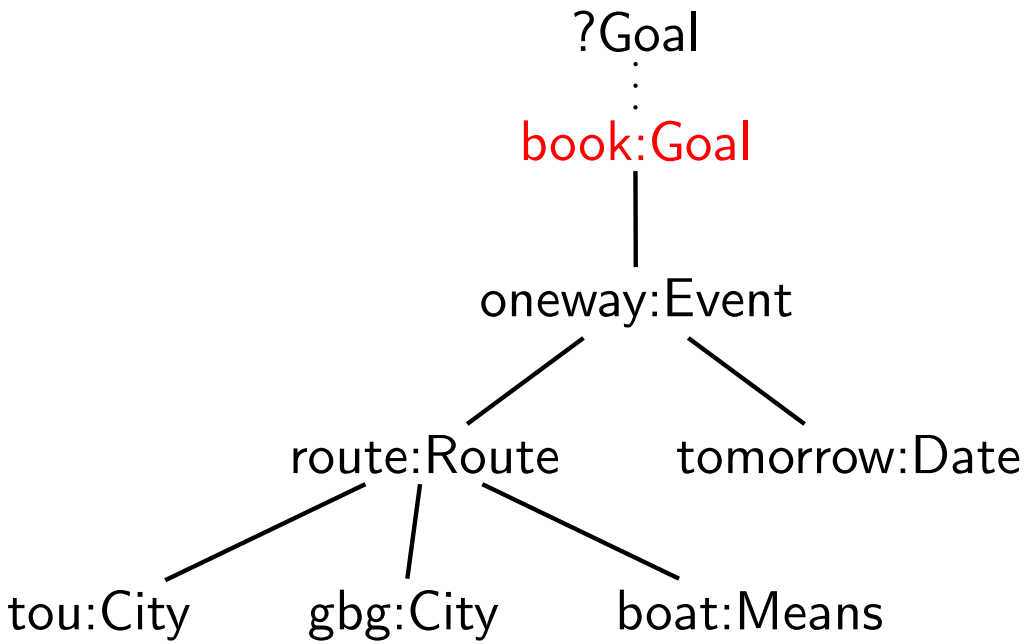
Strategy 3: "Bottom-down" refinement



The question is filtered by the goal type!

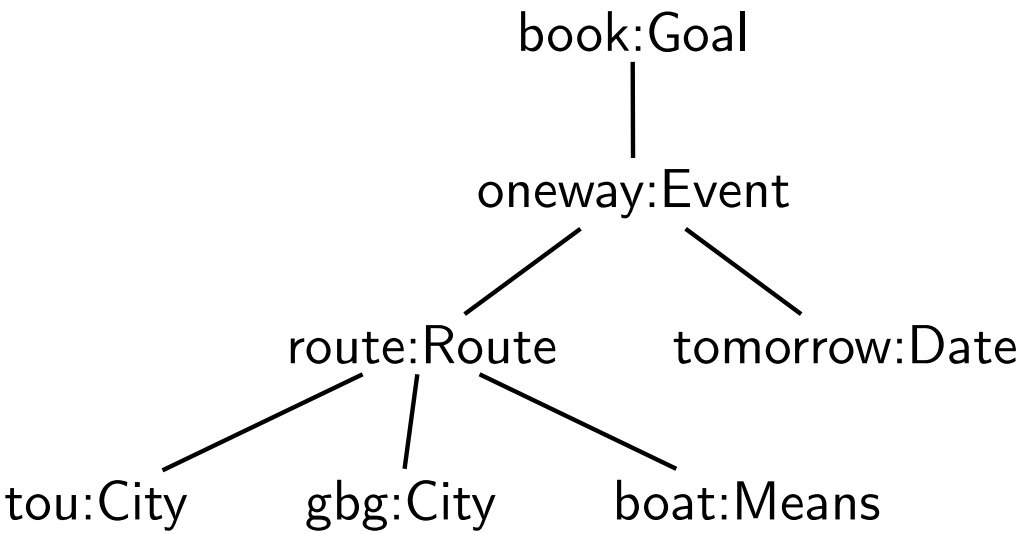
S: "Do you want to book a trip or know the price?"

Strategy 3: "Bottom-down" refinement



U: "Book a trip"

Strategy 3: "Bottom-down" refinement



Three different refinement strategies

So, there are (at least) the following refinement strategies:

1. top-down refinement
2. bottom-up refinement
3. "bottom-down" refinement

Of course, these strategies can be combined, e.g.:

- the strategy could depend on the type of the dominating node
- the strategy could depend on the maximum/minimum distance between the dominating and dominated nodes

Conclusion

To summarize:

- The domain of the dialogue system is specified in type theory
- The dialogue manager builds a type-correct term interactively
- Unfixed tree nodes represent underspecified information
- There are at least three different refinement strategies
  - corresponding to different behaviour of the system

Things left unsaid

What have I not discussed?

- Sub-dialogues and anaphoric expressions
  - can be represented by links between nodes in separate terms
- The usage of dependent types
  - instead of *oneway*, *return* one could have:
    $trip : (t : Triptype) \times Route \times Dates(t) \rightarrow Event$
  - every month has different number of days:
    $date : (m : Month) \times Day(m) \rightarrow Date$
- How to perform parsing and generation of utterances
  - one possibility is to use Grammatical Framework to specify a compositional mapping from terms to utterances
  - or any of the 100s of existing approaches to parsing and generation

Things left to say

What have I not solved?

- How can *feedback* be incorporated in this framework?
  - e.g., "I'm sorry, I didn't hear you."
    or, "Did you say Sweden or Switzerland?"
- How do we manage *corrections*, such as deleting or modifying nodes?
  - e.g., "No, I mean Toulouse, not too loose"
    or, "No, I want a return ticket instead"
- And a lot more things, such as
  - how can all this be *implemented*?