From Logic Forms to ASP query answering

by
Marcello Balduccini

May 12, 2005

Outline

$\Rightarrow$ **Introduction**

- Issues Involved
- Approach
- Overall Achievements
- Technical Achievements
- Conclusions

Motivation

- LCC's NLP tool (*NLPt*):

natural language statements
⇓
statements in a formal language (*Logic Forms*)

- TTU's reasoning/query-answering system (*TTUq*):

knowledge encoded in a formal language (*A-Prolog*)
⇓
Q/A involving fairly deep reasoning

Objective

To connect the two systems, and form the basis of a query-answering tool that accepts natural language queries.

NLPt's Output Language

Output of NLPt: sequence of statements of the form

$$\langle word, POS, sense \rangle (p_1, p_2, \ldots)$$

where:

- $word$: word being described;
- POS: part of speech assigned to $word$ (VB, NN, NE, NNC, CC, ...);
- $sense$: sense (from WordNet's database) in which $word$ is used (may be omitted if $word$ has only one sense);
- p_i's: constants denoting statements in some relation with the given one.

Example of NLPt's Language

Text

John took the plane from Paris to Baghdad.

Logic Form

	John_NN(x1)	"John is a noun and is denoted by $x1$"
&	take_VB_11(e1,x1,x2)	"verb take; sense 11; denoted by $e1$; subject is $x1$; object is $x2$"
&	plane_NN_1(x2)	"noun plane; sense 1; denoted by $x2$"
&	from_IN(e1,x3)	"location (type *from*) of $e1$ is $x3$"
&	Paris_NN(x3)	"noun Paris; denoted by $x3$"
&	to_TO(e1,x4)	"location (type *to*) of $e1$ is $x4$"
&	Baghdad_NN(x4)	"noun Baghdad; denoted by $x4$"

WordNet and eXtended WordNet

Collections of classifications of words (nouns, verbs, adjectives, adverbs), including:

- description of meaning;
- examples of use.

<u>WordNet</u>

- Information is mainly unstructured, in natural language.

<u>eXtended WordNet</u>

- Information extracted from WordNet, structured using XML;
- examples encoded using logic forms, parse trees.

TTUq's Input Language

Collection of statements of the form:

- $h(P, S)$: property P holds at step S of the story, e.g.

$$h(at(john, paris), 0).$$

- $o(A, S)$: action A occurs at step S of the story, e.g.

$$o(go\_on(john, trip(paris, baghdad)), 0).$$

- $rel(p_1, \ldots, p_n)$: objects $p_1, \ldots, p_n$ are in relation rel, e.g.

$$destination(trip(paris, baghdad), baghdad).$$

Outline

- Introduction

⇒ **Issues Involved**

- Approach
- Overall Achievements
- Technical Achievements
- Conclusions

Issues

1. **The languages are different in nature**
   - ◇ NLPt's: statements about *parts of speech*; temporal sequence of events is *implicit*.
   - ◇ TTUq's: statements about *objects, their properties/relations, and about occurrence of actions*; temporal sequence of events is *explicit*.

Example. "The train is at the station. The train is number 176. Number 176 is scheduled to depart at 10:30am."

- NLPt: sentence 1 is about NN "train"; sentence 3 is about NNC "number 176."
- TTUq: sentences should be about the <u>same</u> *object*: "train" / "number 176."

Issues (cont'd)

2. **Different role of knowledge**
   - ◇ Understanding output of NLPt requires extra, *implicit*, domain knowledge.
   - ◇ TTUq requires all relevant knowledge to be encoded.

Example. "John was on the train that arrived at NYC at 1:30pm. Was he at the station at 1:31pm?"

- NLPt: sentence 1 is about "NYC"; sentence 2 is about "the station."
- TTUq: proper answer requires (at least) knowledge that, normally, trains stop at a town's station.

Part of this knowledge can be extracted from the WordNet and eXtended WordNet databases.

Issues (cont'd)

3. **Words with multiple meanings**

Example

"John took the plane from Paris to Baghdad."

- ◇ "take_VB_11" means "going on a trip."
- ◇ Object of "take_VB_11" is *means of transportation*.
- ◇ Mapping from "parameters" of a verb to TTUq's relations entirely depends on the meaning of the verb.

Specific knowledge about take_VB_11 is necessary for the mapping.

Outline

- Introduction
- Issues Involved
- ⇒ **Approach**
- Overall Achievements
- Technical Achievements
- Conclusions

Logic Forms to A-Prolog Translator

Query in natural language

$\Downarrow$

Natural language parsing from LCC

$\Downarrow$

General Translation Algorithm (G2L)
$+$
Declarative Knowledge

$\Downarrow$

Q/A with TTU's reasoning system

$\Downarrow$

Answer

General Translation Algorithm (G2L)

General transformations to make statements suitable for use with action theories, e.g.:

- verbs ⇒ events or properties;
- translation of "identity" statements (e.g. "the train is an Acela-Express 176");
- prepositions ⇒ event's parameters;
- generation of default temporal sequence of events;
- identification and translation of the question.

Declarative Knowledge

- Events $\Rightarrow$ $o(A, S)$ statements.
- Properties $\Rightarrow$ $h(P, S)$ statements.
- Event's params $\Rightarrow$ $rel(p_1, \ldots, p_n)$ statements.
- Domain-independent knowledge:
  - ◇ links events involving objects that are identical;
  - ◇ distinguishes between date/time specifications and space specifications.
- Domain-dependent knowledge:
  - ◇ meaning of events and of their parameters, e.g.
    take_VB_11 ⇒ action go_on;
  - ◇ adjustment of the temporal sequence produced by G2L.

The Translator In Action

Sentence
John took the plane from Paris to Baghdad.

G2L produces:

```
event(e1,take,11).
happened(e1).
raw_event_actor(e1,john).
raw_object(e1,plane).
raw_parameter(e1,from,1,paris).
raw_parameter(e1,to,1,baghdad).
suggested_step(e1, 0).
```

Domain Knowledge Needed (partial):

```
o(go_on(ACT,Obj),S) ←
        event(E,take,11),
        happened(E),
        event_actor(E,ACT),
        object(E,Obj),
        suggested_step(E,S).

dest(Obj,DEST) ←
        event(E,take,11),
        happened(E),
        parameter(E,to,1,DEST),
        object(E,Obj).
```

Outline

- Introduction
- Issues Involved
- Approach

⇒ **Overall Achievements**

- Technical Achievements
- Conclusions

Summary

Test queries

- Performed: translation and query-answering.
- Result: correct answers obtained for all queries.

Steve's travel-related query (travel-1)

- Performed: translation and Q/A for a simplified query.
- Result: correct answers obtained.

Test Queries

- John took the plane from Paris to Baghdad. Is John in Baghdad?
- John took the plane from Paris to Baghdad. The plane has scheduled intermediate stops in Berlin and Rome. Is John in Baghdad?
- **(Variant of above)** John took the plane from Paris to Baghdad. The plane has scheduled intermediate stops in Berlin and Rome. *Where is* John?
- John is in Paris. John packs John's laptop in John's carry-on luggage and takes a plane to Baghdad. Where is John's laptop?

Steve's (Simplified) Query

- The train stood at the Amtrak station in Washington DC (at 10:00am on March 15, 2005).
- The station was Union Station.
- The train was an Acela-Express 176.
- The 176 was scheduled to depart for NYC (at 10:30am) and arrive (at 1:30pm on March 15).
- John arrived by taxi at Union Station (at 10:15am).
- John boarded the Acela-Express (at 10:20am) and handed the train ticket to the conductor.
- The conductor punched the ticket.
- John sat by the window.
- The train left the station on time.

Where is John at the end of the trip?

Difficulties in Steve's Query

- Frequent use of compound nouns, e.g. "Acela-Express 176."
- Parts of compound nouns use to denote the whole noun, e.g. "Acela-Express" instead of "Acela-Express 176."
- Multiple nouns denote the same object, e.g. "train", "Acela-Express", "176."
- Combination of verb "scheduled" with two verbs, whose arguments describe the trip, e.g. "scheduled to depart for NYC..."
- Use of locations and *sub-locations*, e.g. "*at the Amtrak station* in Washington DC."

Outline

- Introduction
- Issues Involved
- Approach
- Overall Achievements

⇒ **Technical Achievements**

- Conclusions

Summary

Translation of

- questions "is [...] at/in [location]?", "where is ...?";
- possessive relation (e.g. "John's laptop");
- phrases used as objects (verb "to schedule" only), e.g. "the train was scheduled to depart [...] and to arrive [...]";
- conjunction, when used with prepositions and in "subordinate phrases" (see previous item);
- verbs denoting identity, e.g. "the train is an Acela-Express 176";
- compound nouns (NNC) used in objects of verbs denoting identity.

Summary (cont'd)

Interpretation of meaning

- ability to preserve sentence connection, e.g. "John took the plane from Paris to Baghdad" and "the plane has scheduled stops in Berlin and Rome";
- extension of sentence connection by using identity relations, e.g. "The train is at the station. The train is the 176. The 176 is scheduled to depart at 10:30am";
- filtering of date/time specs when looking for space specs.
- encoding of the meaning of a substantial number of verbs (be_VB_2, be_VB_3, take_VB_11, pack_VB_1, schedule_VB_1,...).

G2L-Translation of Query 1

"John took the plane from Paris to Baghdad. Is John in Baghdad?"

```
event(e1,take,11).
happened(e1).
raw_event_actor(e1,john).
raw_object(e1,plane).
raw_parameter(e1,from,1,paris).
raw_parameter(e1,to,1,baghdad).
suggested_step(e1, 0).
```

```
answer_true(q(e1)) ←
        h(at(john, baghdad), n).

answer_false(q(e1)) ←
        ¬h(at(john, baghdad), n).

type_query(q(e1), boolean).
```

G2L-Translation of Query 2

"John took the plane from Paris to Baghdad. The plane has scheduled intermediate stops in Berlin and Rome. Is John in Baghdad?"

```
event(e1,take,11).
happened(e1).
raw_event_actor(e1,john).
raw_object(e1,plane).
raw_parameter(e1,from,1,paris).
raw_parameter(e1,to,1,baghdad).

event(e2,schedule,1).
happened(e2).
raw_event_actor(e2,plane).
raw_object(e2,stop).
raw_parameter(stop,in,1,berlin).
raw_parameter(stop,in,2,rome).
```

```
next(e1, e2).
suggested_step(e1, 0).
suggested_step(e2, 1).

answer_true(q(e1)) ←
        h(at(john, baghdad), n).

answer_false(q(e1)) ←
        ¬h(at(john, baghdad), n).

type_query(q(e1), boolean).
```

G2L-Translation of Query 3

"John took the plane from Paris to Baghdad. The plane has scheduled intermediate stops in Berlin and Rome. Where is John?"

```
event(e1,take,11).
happened(e1).
raw_event_actor(e1,john).
raw_object(e1,plane).
raw_parameter(e1,from,1,paris).
raw_parameter(e1,to,1,baghdad).

event(e2,schedule,1).
happened(e2).
raw_event_actor(e2,plane).
raw_object(e2,stop).
raw_parameter(stop,in,1,berlin).
raw_parameter(stop,in,2,rome).
```

```
next(e1, e2).
suggested_step(e1, 0).
suggested_step(e2, 1).

answer_true(q1(C)) ←
        h(at(john, C), n).

type_query(q1(C), find).
```

G2L-Translation of Query 4

"John is in Paris. John packs John's laptop in John's carry-on luggage and takes a plane to Baghdad. Where is John's laptop?"

```
h_at(at(john, paris), e1).

event(e2,pack,1).
happened(e2).
raw_event_actor(e2,john).
raw_object(e2,laptop(john)).
raw_parameter(e2,in,1,lugg(john)).

event(e3,take,11).
happened(e3).
raw_event_actor(e3,john).
raw_object(e3,plane).
raw_parameter(e3,to,1,baghdad).
```

```
next(e1, e2).
next(e2, e3).
suggested_step(e1, 0).
suggested_step(e2, 1).
suggested_step(e3, 2).

answer_true(q1(C)) ←
        h(at(laptop(john), C), n).

type_query(q1(C), find).
```

G2L-Translation of Steve's Query

```
event(e1,stand,3).
happened(e1).
raw_event_actor(e1,train).
raw_parameter(e1,in,1,washdc).
raw_parameter(e1,at,1,station).
raw_parameter(e1,at,2,n1000am).
raw_parameter(e1,on,1,n031505).

event(e3,schedule,1).
happened(e3).
raw_event_actor(e3,number176).
subordinate(raw_object,e3,e4).
subordinate(raw_object,e3,e5).

event(e4,depart,1).
happened(e4).
raw_event_actor(e4,number176).
raw_parameter(e4,at,1,n1030am).
raw_parameter(e4,for,1,nyc).

event(e5,arrive,1).
happened(e5).
raw_event_actor(e5,number176).
raw_parameter(e5,at,1,n130pm).
raw_parameter(e5,on,1,mar15).

same(unstat, station).
same(union, station).
same(station, station).

same(acela176, train).
same(acelaexpress, train).
same(number176, train).

answer_true(q1(C)) ←
        h(at(john, C), n).

type_query(q1(C), find).
```

Domain-Independent Knowledge

```
%% raw_XXX: produced by G2L from output of NLPt

event_actor(E,ACTOR) :-
        raw_event_actor(E,ACTOR'),
        same_as(ACTOR',ACTOR).

object(E,Obj) :-
        raw_object(E,Obj'),
        same_as(Obj',Obj).

parameter(E,TYPE,NUM,PARM) :-
        raw_parameter(E,TYPE,NUM,PARM'),
        same_as(PARM',PARM).
```

%% Definition of same_as (*simplified*)
%% [transitive closure of relation "same"]

```
same_as(X,Y) :-
        same(X,Y).

same_as(X,Y) :-
        same(X,Z),
        same_as(Z,Y).
```

Domain Knowledge: Going on Trips

```
o(go_on(ACTOR,Obj),STEP) ←
        event(E,take,11),
        happened(E),
        event_actor(E,ACTOR),
        object(E,Obj),
        suggested_step(E,STEP).

trip(Obj) ←
        event(E,take,11),
        happened(E),
        object(E,Obj).

h(trip_by(Obj,Obj),STEP) ←
        event(E,take,11),
        happened(E),
        object(E,Obj),
        suggested_step(E,STEP).

actor(go_on(ACT,Obj),ACT) ←
        event(E,take,11),
        happened(E),
        event_actor(E,ACT),
        object(E,Obj).

origin(Obj,ORIG) ←
        event(E,take,11),
        happened(E),
        parameter(E,from,1,ORIG),
        object(E,Obj).

dest(Obj,DEST) ←
        event(E,take,11),
        happened(E),
        parameter(E,to,1,DEST),
        object(E,Obj).
```

Going on Trips (cont'd)

```
leg_of(Trip,ORIGIN,LEG1) ←
    event(E,schedule,1),
    happened(E), object(E,stop),
    event_actor(E,Trip),
    origin(Trip,ORIGIN),
    parameter(stop,in,1,LEG1).

leg_of(Trip,LEGA,LEGB) ←
    event(E,schedule,1),
    happened(E), object(E,stop),
    event_actor(E,Trip),
    parameter(stop,in,N,LEGA),
    parameter(stop,in,N+1,LEGB).

leg_of(Trip,LEGN,DEST) ←
    event(E,schedule,1),
    happened(E), object(E,stop),
    event_actor(E,Trip),
    num_stops(Trip,N),
    parameter(stop,in,N,LEGN),
    dest(Trip,DEST).

num_stops(Trip,N) ←
    has_atleast(Trip,N),
    not has_atleast(Trip,N+1).

has_atleast(Trip,N+1) ←
    event(E,schedule,1),
    happened(E), object(E,stop),
    event_actor(E,Trip),
    parameter(stop,in,N,LEGA),
    parameter(stop,in,N+1,LEGB).
```

Domain Knowledge: Misc

%% Knowledge about Packing

```
o(pack(ACTOR,WHAT,WHERE),STEP) ←
        event(E,pack,1),
        happened(E),
        event_actor(E,ACTOR),
        object(E,WHAT),
        parameter(E,in,1,WHERE),
        suggested_step(E,STEP).

actor(pack(ACTOR,WHAT,WHERE),ACTOR) ←
        event(E,take,11),
        happened(E),
        event_actor(E,ACTOR),
        object(E,WHAT),
        parameter(E,in,1,WHERE).
```

%% Properties of objects (Events ⇒ Steps)

```
h(Fl,STEP) ←
        h_at(Fl,E),
        suggested_step(E,STEP).
```

General Knowledge about Queries

```
%% Knowledge about answering boolean queries
%
yes ←
        type_query(Q,boolean),
        answer_true(Q).

no ←
        type_query(Q,boolean),
        answer_false(Q).

maybe ←
        type_query(Q,boolean),
        not yes,
        not no.

%% Knowledge about answering "find" queries
%
ans(Q) ←
        type_query(Q,find),
        answer_true(Q).
```

Outline

- Introduction
- Issues Involved
- Approach
- Overall Achievements
- Technical Achievements

⇒ **Conclusions**

Usage of Translator and Q/A System

- Split logic form of paragraph in *affirmative part* and *question*.
- Store affirmative part in file "test1".
- Store question in file "query1".
- Run:
  translate.sh > test1.apl *(on Unix)*
  This invokes G2L and stores the output in file "test1.apl".
- Run:
  lparse –true-negation -c n=7 test1.apl interface.pl travel | smodels | mkatoms
  This runs the inference engine and outputs the answer to the query. *File "interface.pl" contains the Declarative Knowledge necessary to complete the translation.*

Output for the queries

- "John took the plane from Paris to Baghdad. Is John in Baghdad?"

  Answer: **yes**
  ::endmodel

- "John took the plane from Paris to Baghdad. The plane has scheduled intermediate stops in Berlin and Rome. Is John in Baghdad?"

  Answer: **yes**
  ::endmodel

- "John took the plane from Paris to Baghdad. The plane has scheduled intermediate stops in Berlin and Rome. Where is John?"

  Answer: **ans(q1(baghdad))**
  ::endmodel

- "John is in Paris. John packs John's laptop in John's carry-on luggage and takes a plane to Baghdad. Where is John's laptop?"

  Answer: **ans(q1(baghdad))**
  ::endmodel

- Steve's query: "where is John?"

  Answer: **ans(q1(nyc))**
  ::endmodel

Future Work

- Connection with other research work to allow reasoning about time, intentions and queries involving planning.
- Computation of the answer to Steve's original query.
- More general translation of conjunction and compound nouns.
- Encoding of the meaning of more verbs.