On Representation in TRIPS

George Ferguson

28 Feb 2007

AKRL Motivation

- Interlingua between the linguistic processing and back-end reasoning systems, which vary from application to application
- Retains some aspects of natural language that must be handled by the reasoning systems themselves, or handled by some preprocessing layer before the main reasoners are invoked.

AKRL Design Principles

- Frame-like representation
- Uses a domain-specific ontology
- Ontology mapping rules to translate the semantics of natural language to and from the target ontology
- AKRL syntax remains the same across domains
- Allows us to represent those aspects of meaning necessary to support natural language interpretation and connect it to task- and domain-specific reasoning

AKRL Syntax

- AKRL terms denote objects:
  (indicator symbol attribute value attribute value ...)

- "a green robot"

```
(A V1 :instance-of ROBOT :color GREEN)
```

- "a robot more than six feet tall"

```
(A V1 :instance-of ROBOT :height H1)
(A H1 :instance-of HEIGHT-QUANTITY
      :unit FEET :number N1)
 (A N1 :instance-of NUMBER :min-value 6)
```

AKRL Ontologies

- Attributes (roles) and their values come from underlying onotologies

  `(A V1 :instance-of ROBOT :color GREEN)`

  `ROBOT` is an ontological type

  `color` is a property of `ROBOT`s

  `GREEN` is value for the `color` of `ROBOT`s

- AKRL specifies additional roles for various purposes (e.g., definite reference), as well as for "built-in" type of objects (e.g., sets)

AKRL Semantics

- AKRL term corresponds to set of subject-verb-object assertions:

```
(A V1 :instance-of ROBOT :color GREEN)
```

```
(V1 instance-of ROBOT)
(V1 color GREEN)
```

- AKRL symbols are globally defined

AKRL Indicators

- `A`: Indefinite reference
- `THE`: Definite reference
- `KIND`: Complex type expression
- `RELN`: Relational entity, such as actions, events, and properties
- `QUANT`: Quantified expression

AKRL Sets

- Sets are ubiquitous in NL and are generally treated idiosyncratically in KR languages
- Use object type `SET`
- Attributes: `element-type`, `size`, `subset-of`, `members`, `not-members`
- "the three robots"

```
(THE V40 :instance-of SET :element-type ROBOT :size 3)
```

- "three of the trucks"

```
(A V43 :instance-of SET :size 3 :subset-of V44)
(THE V44 :instance-of SET :element-type TRUCK)
```

AKRL Complex Types

- Complex types are defined using the `KIND` indicator

```
(KIND K1 :instance-of ROBOT :color GREEN)
```

- Can then be used wherever a type can be used:

  "the green robots"

```
(THE S1 :instance-of SET :element-type K1)
```

AKRL Sequences

- Sequences are ordered sets
- Use object type SEQ
- Attributes: order, direction

```
(A S1 :instance-of SEQ :element-type COMPUTER
      :order COST :direction LESS-THAN)
```

- SEQ from SET:

```
(A S3 :instance-of SET :element-type
  COMPUTER)
(THE S4 :instance-of SEQ :set S3
  :order COST  :direction LESS-THAN)
```

AKRL Sequence Operators

- Attribute: `select`, operators: `MIN`, `MAX`, `NTH`

- "the cheapest computers"

```
(THE S5 :instance-of SET :select (MIN S1))
```

- "the five cheapest computers"

```
(THE S7 :instance-of SET :select (MIN S1 :count 5))
```

- "five of the cheapest computers"

```
(A S8 :instance-of SET :select (MIN S1 :count 5))
```

- "the cheapest computer"

```
(THE C1 :instance-of COMPUTER :select (MIN S1
  :count 1))
```

- "the second cheapest computer"

```
(THE C2 :instance-of COMPUTER :select (NTH S1
  :index 2))
```

AKRL Set/Sequence Examples

- "the cheapest, lightest black computers"

```
(KIND K1 :instance-of COMPUTER :color BLACK)
(THE S1 :instance-of SEQ :element-type K1 :order
  COST :direction LESS-THAN)
(THE S2 :instance-of SET :select (MIN S2))
(THE S3 :instance-of SEQ :set S2 :order WEIGHT
  :direction LESS-THAN)
(THE C1 :instance-of K1 :select (MIN S3 :count 1))
```

- "two of the five cheapest computers"

```
(KIND K1 :instance-of COMPUTER)
(THE S1 :instance-of SEQ :element-type K1 :order
  COST :direction LESS-THAN)
(THE S2 :instance-of SET :select (MIN S2 :count 5))
(THE S3 :instance-of SET :subset-of S2 :size 2)
```

AKRL Relationals

- Terms described using RELN correspond to reified sentential or propositional entities

  "John believes the truck is green"

```
(RELN V52 :instance-of BELIEVE :agent V53
  :belief V54)
(THE V53 :instance-of PERSON :name John)
(RELN V54 :instance-of ROLE-VALUE :role HAS-
  COLOR :object V55 :value V56)
(THE V56 :instance-of TRUCK)
(THE V57 :instance-of COLOR :equals GREEN)
```

- "Built-in" `RELN ROLE-VALUE` reifies property names

AKRL Quantifiers

- Quantifiers are very complex in NL
- AKRL provides two mechanisms
  - Quantity quantifiers:
    - Have a set size interpretation require a plural head noun (e.g., *"several trucks"*)
    - Cannot take a numeric modifier (e.g., * *"several three trucks"*).
  - Logical quantifiers
    - Like generalized first-order logic quantifiers

AKRL Quantity Quantifiers

- *several*, *many*, *a few*, *not many*, ...
- Interpreted as `SET`s with the quantifier as the `:size` of the set:

  "several trucks"

```
(A V60 :instance-of SET :element-type TRUCK
  :size SEVERAL)
```

  "at least three trucks"

```
(A V61 :instance-of SET :element-type TRUCK
  :size V62)
(A V62 :instance-of NUMBER :min-value 3)
```

AKRL Logical Quantifiers

- Use indicator `QUANT`

  "every truck"

```
(QUANT V70 :instance-of TRUCK :quantifier
  EVERY)
```

- Domain of quantification may be implicit or explicit (from context)

  "every one of the trucks"

```
(QUANT V71 :instance-of TRUCK :quantifier
  EVERY :domain V72)
(THE V73 :instance-of SET :equals KR888)
```

- Special care needed for *all* and *some*