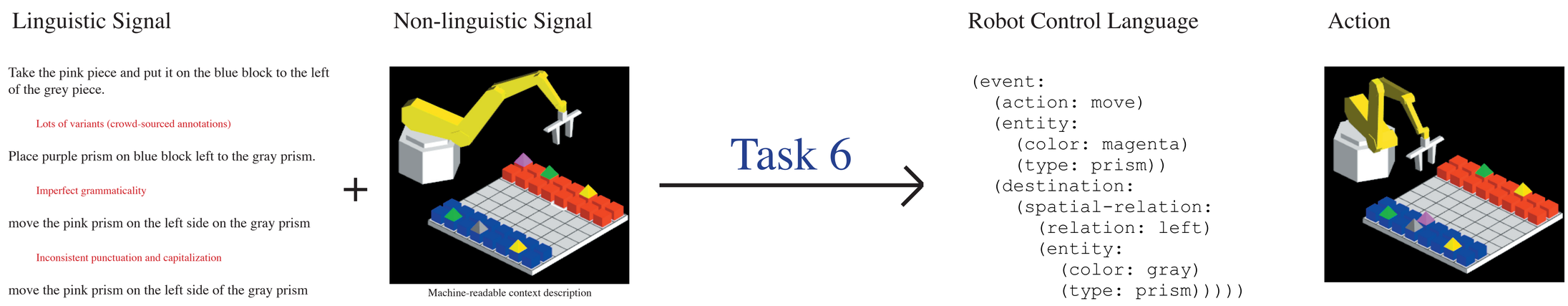
UW-MRS: Leveraging a Deep Grammar for Robotic Spatial Commands

Woodley Packard – University of Washington



High Precision Approach:

Hand-crafted translation rules from an existing hand-crafted grammar

The English Resource Grammar (ERG) is a precision broad coverage grammar of English, embodying roughly 20 person-years of work. It employs Minimal Recursion Semantics to give its analyses detailed predicate argument structure and the scopal relationships that can be accurately ascertained from syntax alone, while remaining underspecified with respect to the relative scoping of quantifiers.

Drop the blue cube.

$$\left(\text{INDEX} = e, \left\{\texttt{pron}(x), \texttt{cube\_n}(y), \texttt{drop\_v\_cause}(e, x, y), \texttt{blue\_a}(\_, y)\right\}\right)$$

Slight modifications required:

– enable the (normally undesirable) rule for determinerless NPs

– a couple of new lexemes for square, tile, etc. used as units of measure (drop it three squares to the left)

Result: ~ 99% grammar coverage on training data

Step 1: Parse with the ERG

Output: a list of MRS structures ranked by the ERG's statistical parse selection model

(MRS$_1$, MRS$_2$, MRS$_3$, …, MRS$_N$)

Step 2: Filter out-of-domain analyses

Since the ERG is a general-purpose tool, we get some domain-inappropriate readings, e.g. block as a verb.

(MRS$_3$, MRS$_7$, MRS$_{13}$, …, MRS$_{N-2}$)

Step 3: Translate MRS to RCL

Hand-written rules (roughly 1,000 lines of C) map MRS structures into RCL trees. The MRS is traversed starting from the INDEX, with top-level conjunctions translating to sequence: elements and main verbs translating to event: elements. Referential indices (x,y) translate into entity: elements, and modifiers translate into color:, type:, spatial-relation:, etc.

The required information (except coreference) is all present and carefully organized in the MRS. Changing the shape of that information from MRS graphs to RCL statements is deterministic and easy.

(RCL$_3$, RCL$_{13}$, RCL$_{29}$, …, RCL$_{N-2}$)

Step 4: Validate with the spatial planner

Candidate RCL statements which are nonsensical given the spatial context are rejected (e.g. trying to drop something when the robot arm is not holding anything, or referencing an entity that does not exist).

(RCL$_{13}$, RCL$_{42}$)

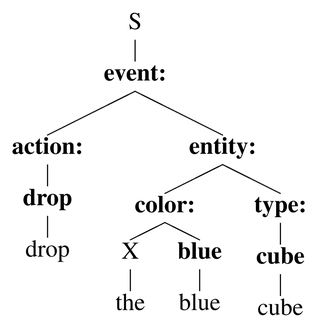
Output: The highest ranked RCL produced by the pipeline (if any) is the result.

RCL$_{13}$

After these steps, roughly 3% of training data items receive no RCL translation, for various reasons. This prompted the investigation of a purely statistical backup system. The robust design payed off in the formal evaluation, where ERG coverage dropped to only 91% (largely due to worse-than-expected vagrant punctuation marks). Precision remained high for the evaluation phase.

High Coverage Approach:

Berkeley parser with a simple (but lossy) transformation



The Berkeley parser tools can learn an LPCFG from collections of phrase structure trees, and then assign ranked phrase structure trees to unseen text, all without manual effort.

The Task 6 training data is a collection of RCL statements with a partial alignment to the underlying text. While RCL statements are trees, the terminals are not in one-to-one correspondence with the tokens of the corresponding utterance; there are deletions (words like the, which RCL considers to be semantically vacuous) and insertions (e.g. id: elements and elided pronouns).

Step 1: Transform the training data to phrase structure trees.

Missing words are inserted into the following constituent with tag X.

(Tree$_1$, Tree$_2$, Tree$_3$, …, Tree$_N$)

Step 2: Train an LPCFG using the Berkeley parser tools.

<LPCFG>

Step 3: Parse an utterance using the LPCFG and the Berkeley parser.

Tree$_1$

Step 4: Drop X nodes from the tree and heuristically insert missing id: elements.

RCL$_1$

Output: The translation procedure always produces exactly one RCL output.

RCL$_1$

Unfortunately, time did not permit exporing the usage of the spatial planner as a filter on an N-best list from the Berkeley parser. This likely would have improved its precision somewhat, but might have reduced its effectiveness as a completely robust fallback (since some items might then have received no result at all).

Results

System	Dev P	Dev R	Eval P	Eval R
MRS-only (−**SP**)	90.7	88.0	92.1	80.3
MRS-only (+**SP**)	95.4	92.2	96.1	82.4
Robust-only (−**SP**)	88.2	88.2	81.5	81.5
Combined (−**SP**)	90.8	90.8	90.5	90.5
Combined (+**SP**)	95.0	95.0	92.5	92.5
ERG coverage		98.6		91.0

References

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, *6*(01), 15-28.

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (pp. 433–440).