

SUMMARY

- **Goal:** Develop a shared task system taking advantage of prior work on surface realization with **OpenCCG** (Espinosa et al., 2008; White and Rajkumar, 2009; *inter alia*)
- **This Year's Strategy:** Try straightforward conversion from deep shared task inputs to "native" OpenCCG inputs, together with techniques for enhanced robustness (see ENLG poster on glue rules for robust chart realization)
- **Lesson:** Input conversion was more difficult than anticipated; these difficulties led us to develop a relation tagger to map shared task inputs to native inputs
- **Results:** Despite these efforts, frequent head inversions and other more complex structural divergences led to a low percentage of grammatically complete realizations, yielding disappointing results
- **Subsequent Analysis:** CCG's **principled approach to relativization and extraction** was the source of a substantial number of input incompatibilities; such divergences **reveal deficiencies in the shared task inputs**
- **Future Work:** We plan to examine whether more comprehensive use of machine learning in input conversion or inducing grammars that are more directly compatible with shared task inputs can achieve high quality results

Background

- OpenCCG is a parsing/realization library for Combinatory Categorical Grammar (Steedman, 2000), a unification-based categorial grammar formalism
- The deep task was chosen as the restriction to dependency trees in the surface task is incompatible with CCG's treatment of coordination, relativization and control (see paper)
- OpenCCG implements a chart realization algorithm in the tradition of Kay (1996); such approaches can potentially deliver very high quality, but achieving broad coverage is a challenge, as is robustness to any deviations in the expected input
- OpenCCG's "native" inputs are derived from gold standard derivations in the CCGbank (Hockenmaier and Steedman, 2007), as is the extracted English grammar; see box at right for details of shared task input conversion
- Realization makes use of an adaptive hypertagging strategy and an averaged perceptron scoring model incorporating *n*-gram and syntactic features
- For robustness, fragments are assembled if no grammatically complete realization can be found

Results

- Development set scores for all realizations (OSU.1) and grammatically complete realizations only (OSU.2) for the shared task inputs and using native inputs are given in the table below
- As the table shows, the percentage of grammatically complete realizations for the converted shared task inputs is well below the percentage using native inputs, with a corresponding drop in BLEU scores (note that even native scores and completeness are somewhat lower than expected, presumably due to a still mysterious drop in hypertagger performance)

System	Shared Task		Native	
	BLEU	5-best Coverage	BLEU	5-best Coverage
OSU.1 (all)	0.4346	0.2483	0.7838	0.5177
OSU.2 (complete)	0.6564	0.3874	0.8341	0.5413

Acknowledgements

This work was supported in part by NSF grant IIS-0812297 and by an allocation of computing time from the Ohio Supercomputer Center.

Input Conversion

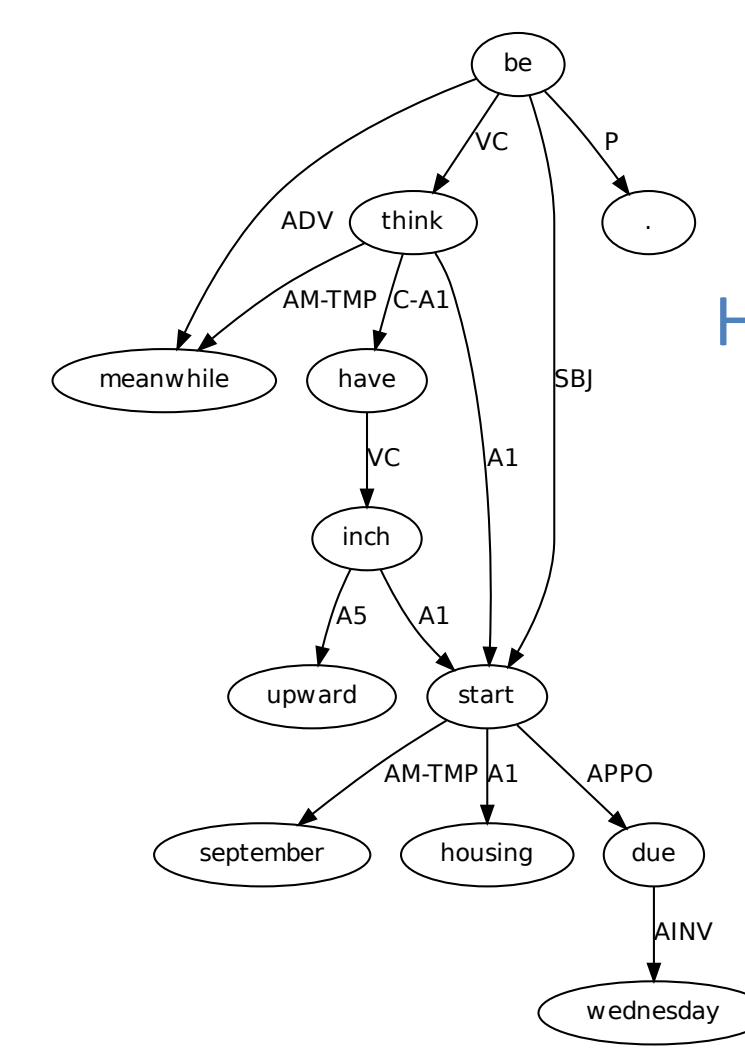
- Conversion rules were written to change the structure of conjunctions, possessives and relative clauses (691, 214 and 90 cases, respectively, in the development set)
- After noticing many **head-dependent mismatches** between the shared task and native inputs, heuristic head inversion rules were implemented (**541 cases** in the devset)
- Named entities and hyphenated words were collapsed into single nodes (703 and 303 cases in the devset, resp.)
- After the initial heuristic conversion, aligned nodes in the partially converted shared task inputs and native inputs were used to train a maxent node POS tagger and relation tagger using the local graph context (devset accuracy of 90% and 90.5%, resp.); a final conversion step heuristically added optional null determiner features and made certain relations optional
- To derive possible word forms for unseen lemmas, **morphg** (Minnen et al., 2001) was used with POS tag heuristics

Examples

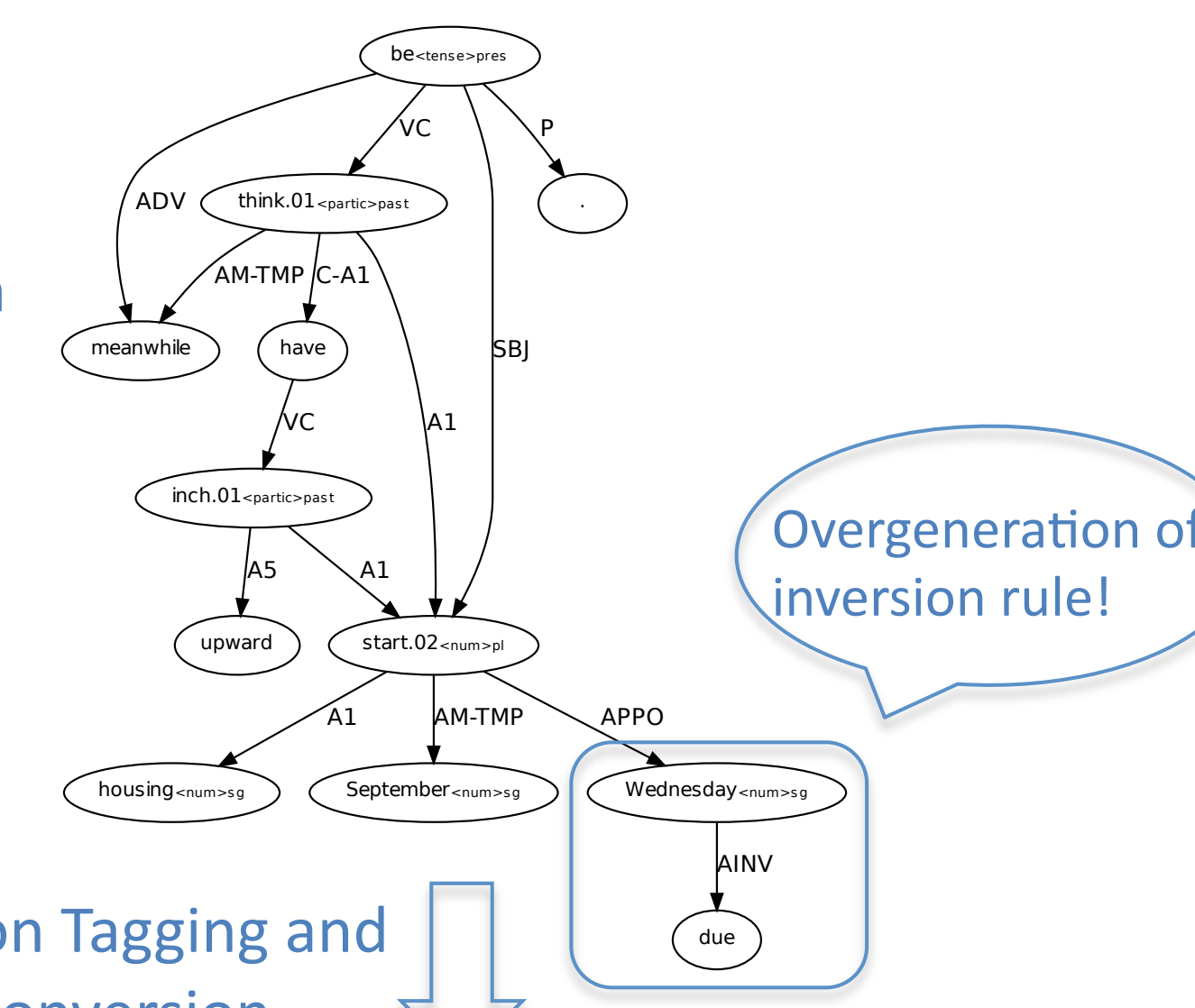
- In the example at right, a heuristic inversion rule overgenerates, causing the desired derivation to fail (a case where the rule succeeds is devset #2, where *due* and *out* are correctly swapped to make *due* the head); here the glue rules nevertheless enable a comprehensible output: *September housing starts, due Wednesday, meanwhile, is thought to have inched upward.*
- The examples below illustrate two **more complex structural divergences** where the shared task inputs are missing crucial semantic dependencies; other problematic cases include extracted *when*-clauses, *tough*-constructions, and comparatives and superlatives

Input Conversion

Shared Task Input

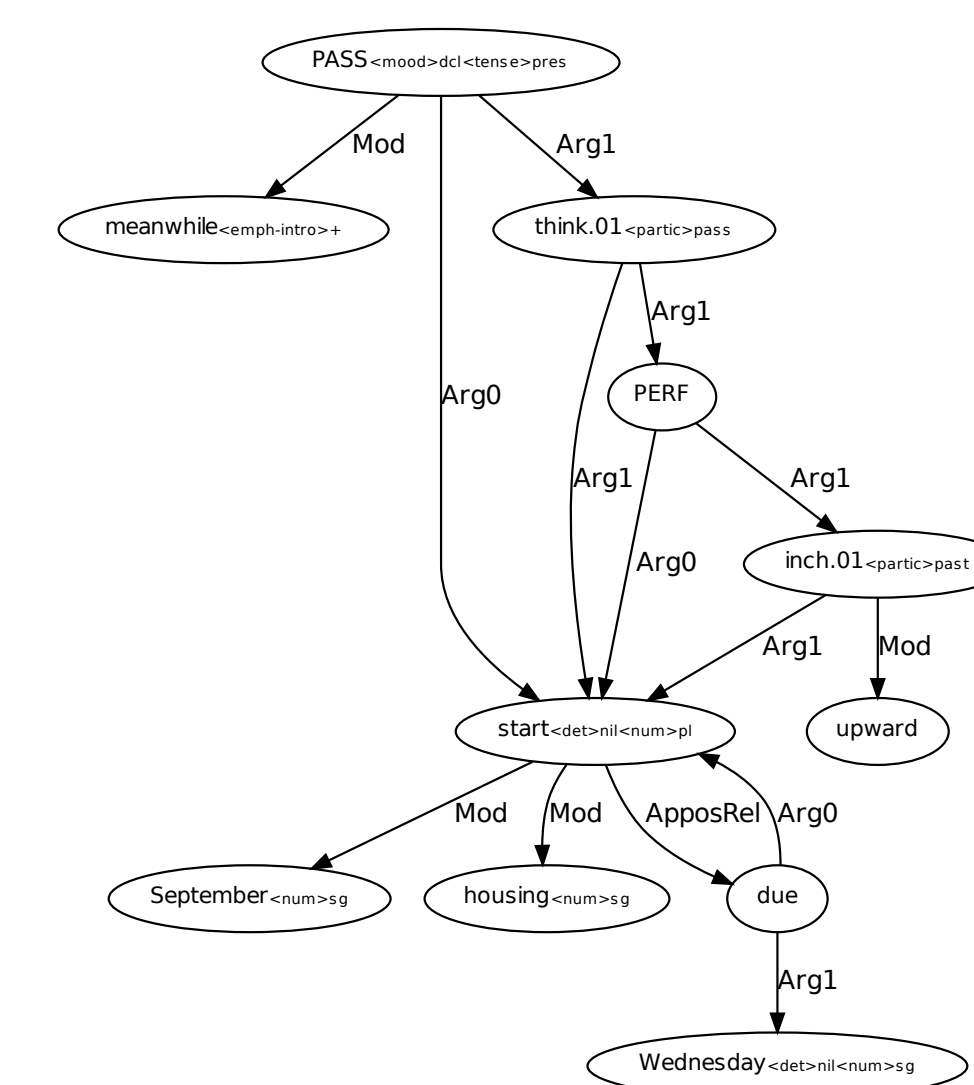


Intermediate Step

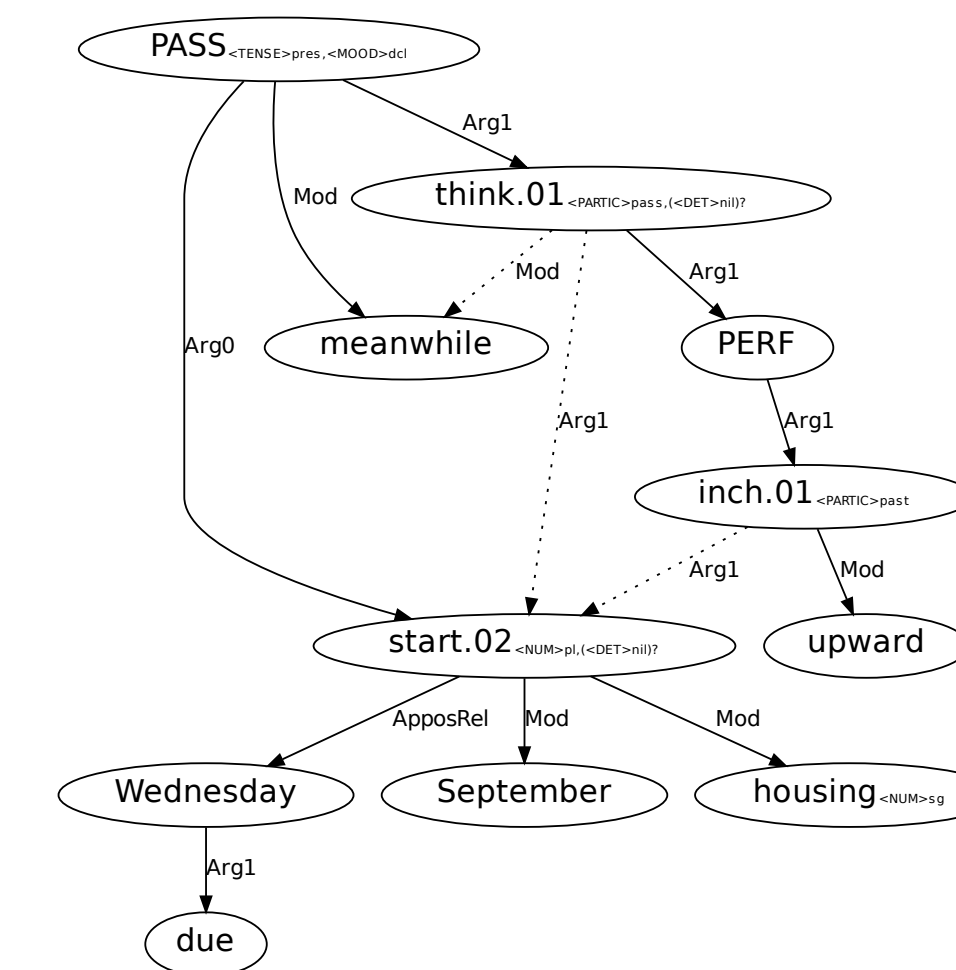


Relation Tagging and Final Conversion

Native Input



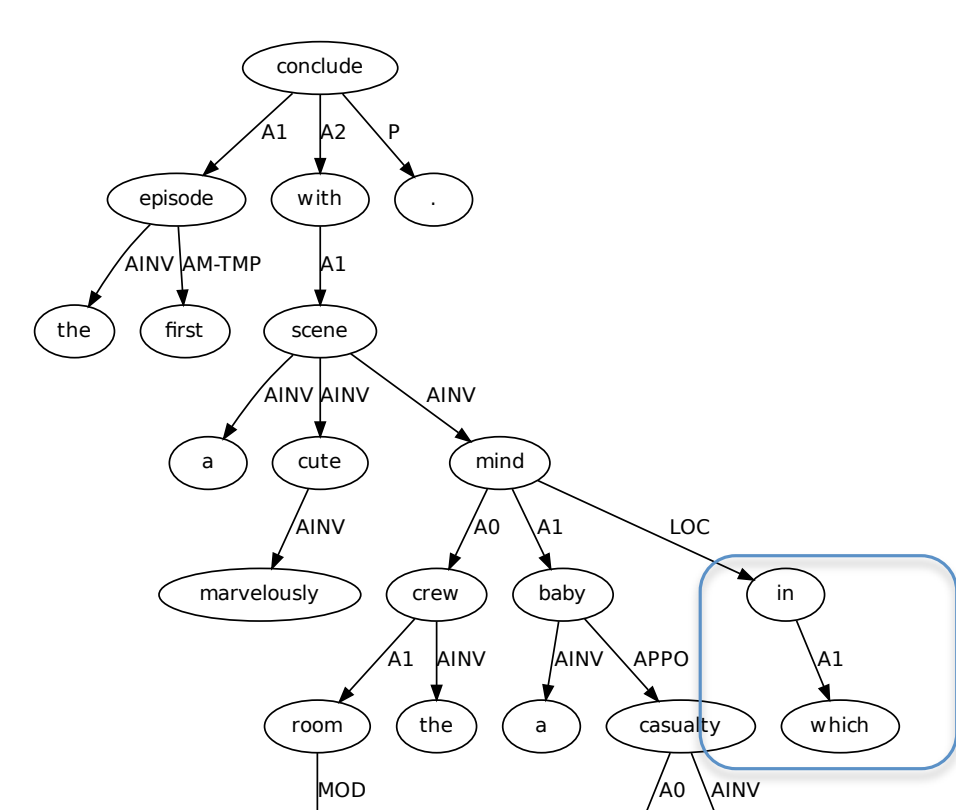
Converted Input



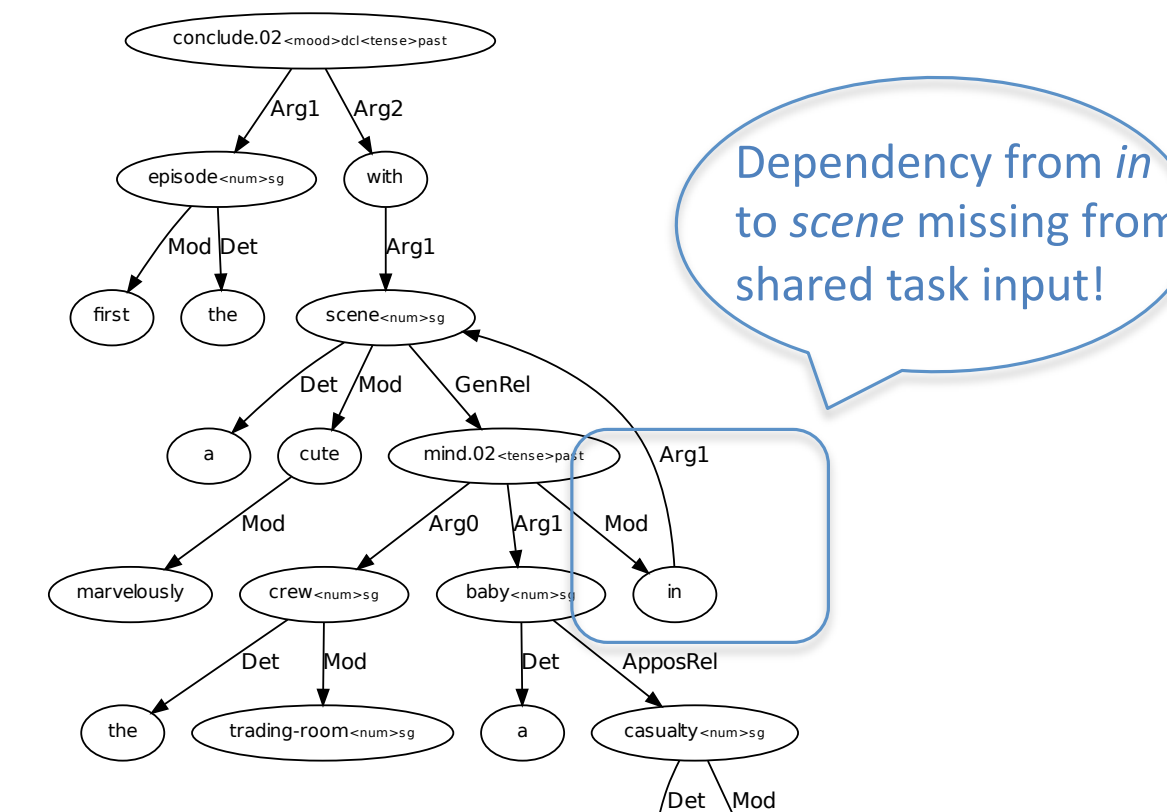
Meanwhile, September housing starts, due Wednesday, are thought to have inched upward. (devset #7)

Structural Divergence Examples: Complex Relativizers and Free Object Relatives

Shared Task Input



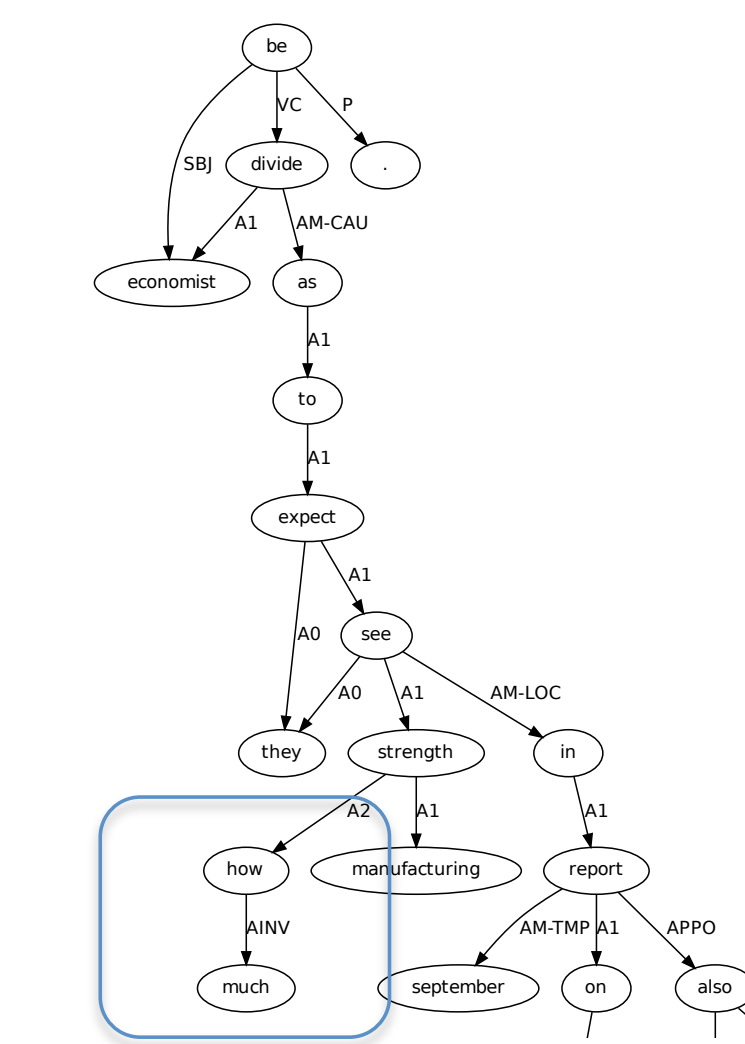
Native Input



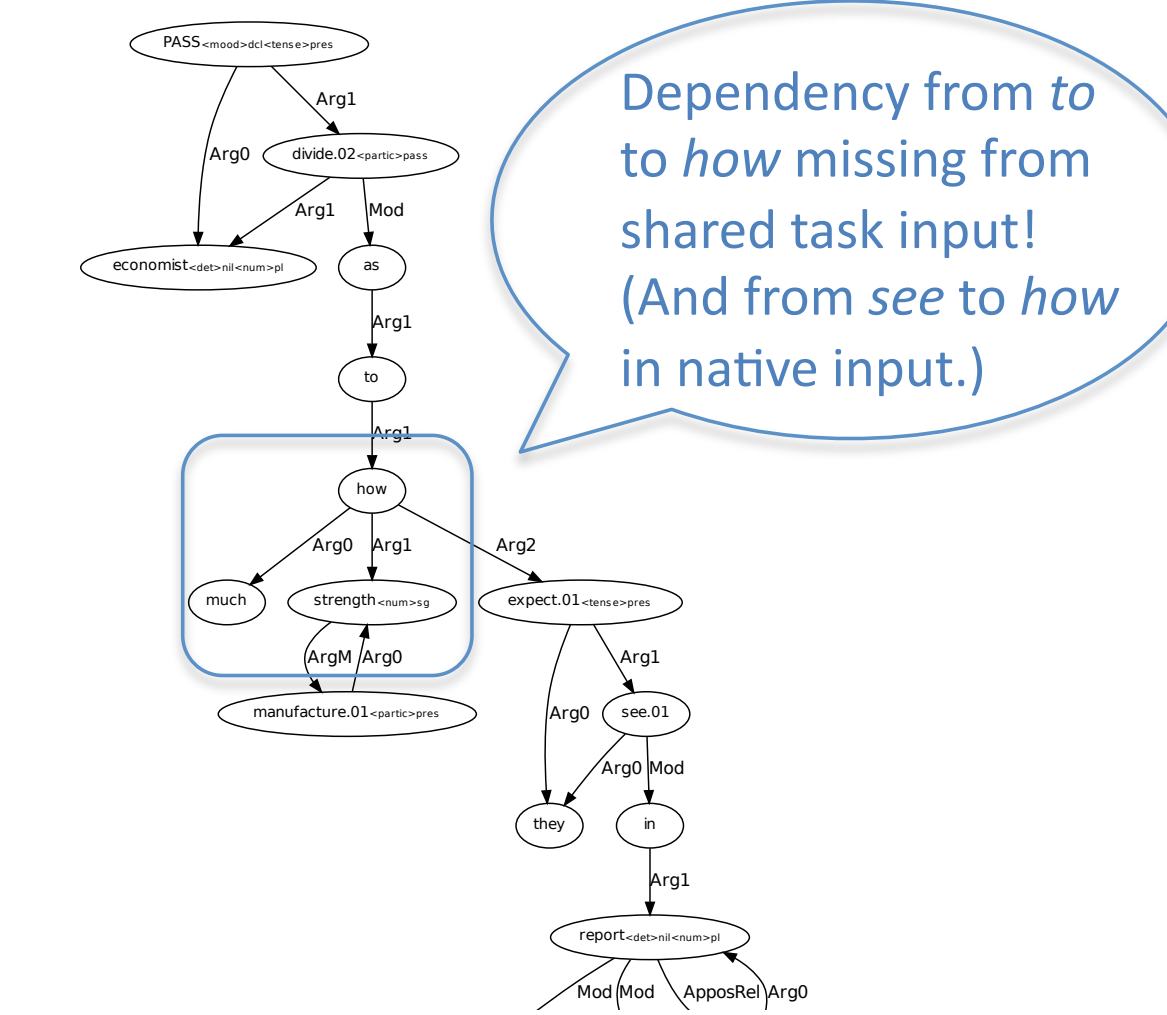
Dependency from *in* to *scene* missing from shared task input!

The first episode concluded with a marvelously cute scene [in which] the trading-room crew minded a baby t_i; the casualty of a broken marriage at the firm. (devset #62)

Shared Task Input



Native Input



Dependency from *to* to *how* missing from shared task input! (And from *see* to *how* in native input.)

Economists are divided as to [how much manufacturing strength] t_i they expect to see t_i in September reports on industrial production and capacity utilization, also due tomorrow. (devset #6)