Generation from AMR using Tree Transducers

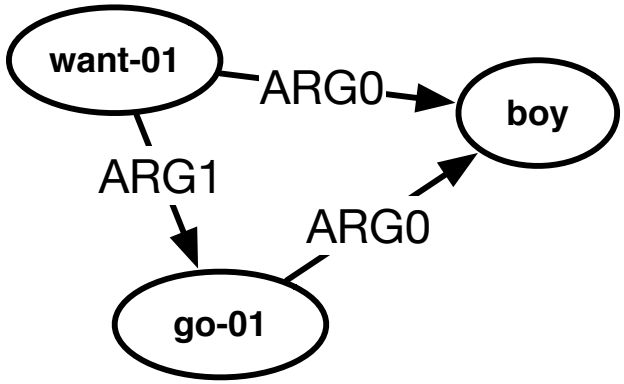
Jeffrey Flanigan

CMU/LTI

Generation from AMR

The boy wants to go.

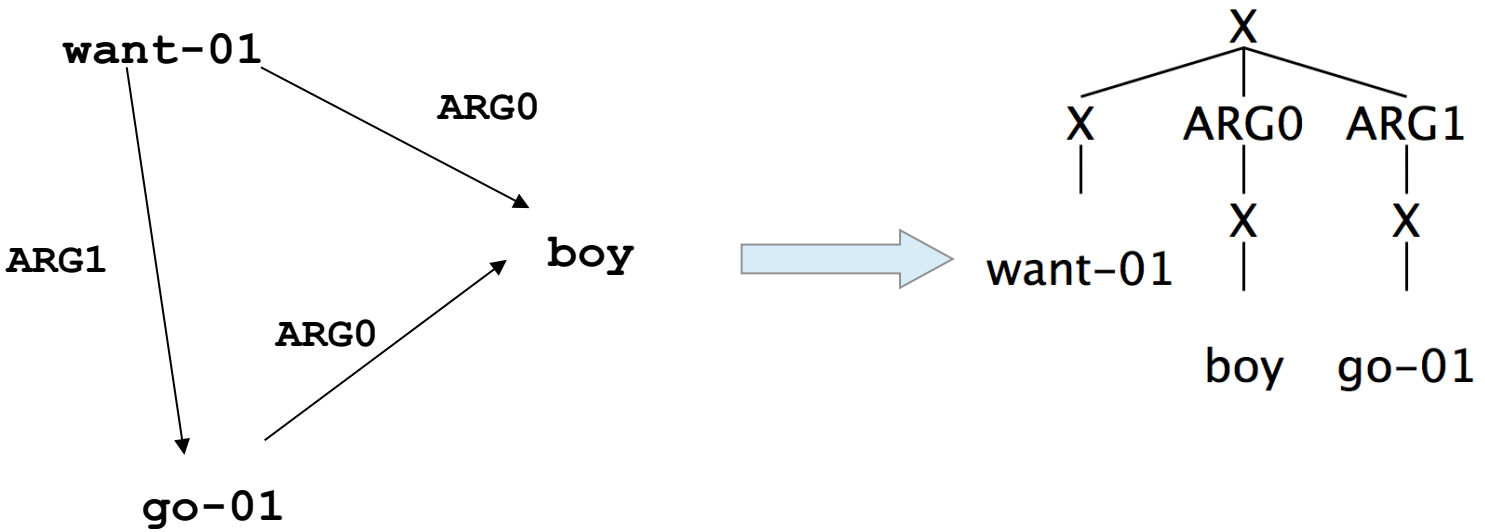
```
(w / want-01
  :ARG0 (b / boy)
  :ARG1 (g / go-01)
          :ARG0 b))
```



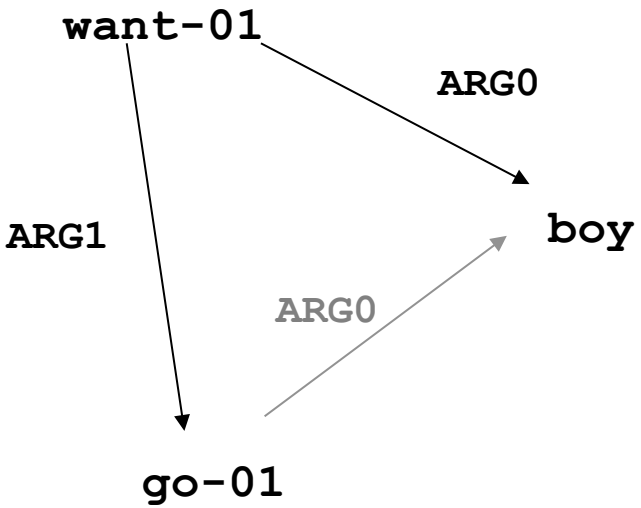
General Approach

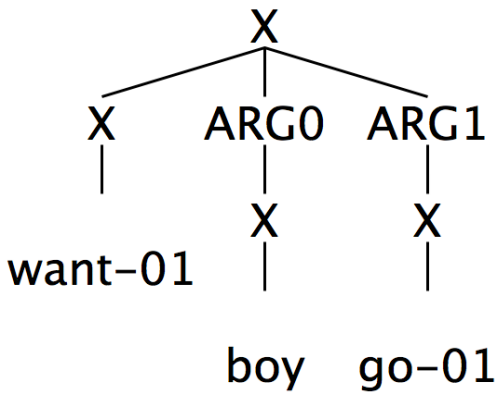
- Compute spanning tree(s)
- Tree transducer rules define output hypergraph
  - Transduction rules can depend on input graph (synthetic rules)
- Search hypergraph for best output
  - Language model
  - Other features as well

Decoding


want-01
ARG0
ARG1
boy
ARG0
go-01
X
X
ARG0
ARG1
X
X
want-01
boy
go-01

Decoding

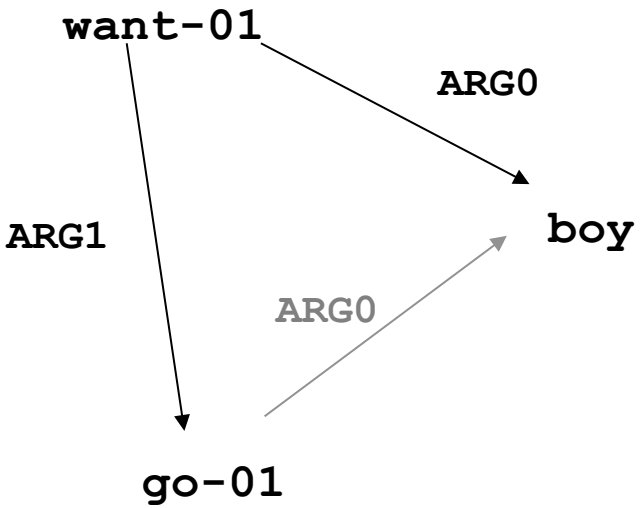


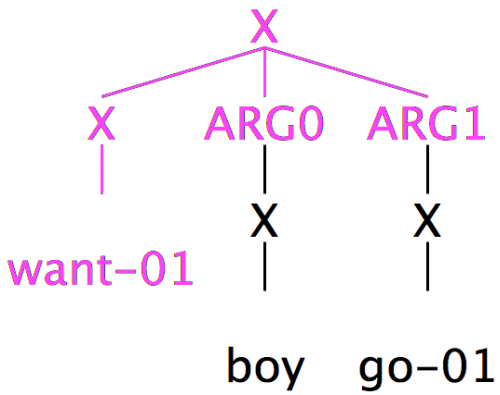


Rule inventory

```
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants a [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants to [2]
(X go-01) ||| go
(X go-01) ||| going
(X boy) ||| the boy
```

Decoding



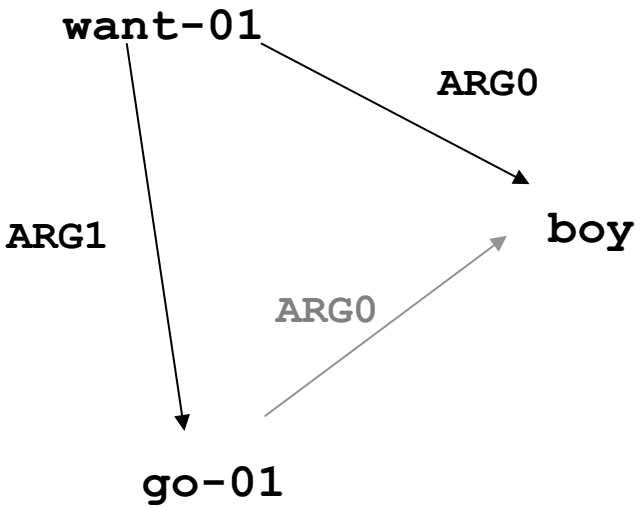


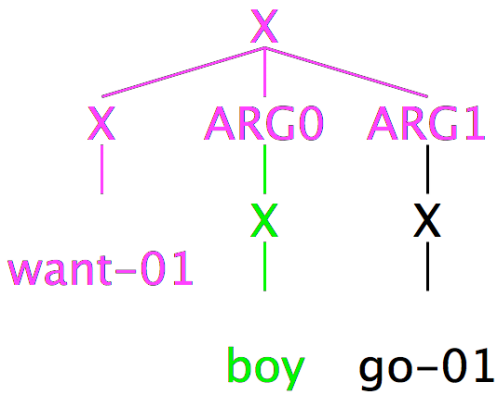
[1] wants to [2]

Rule inventory

```
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants a [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants to [2]
(X go-01) ||| go
(X go-01) ||| going
(X boy) ||| the boy
```

Decoding



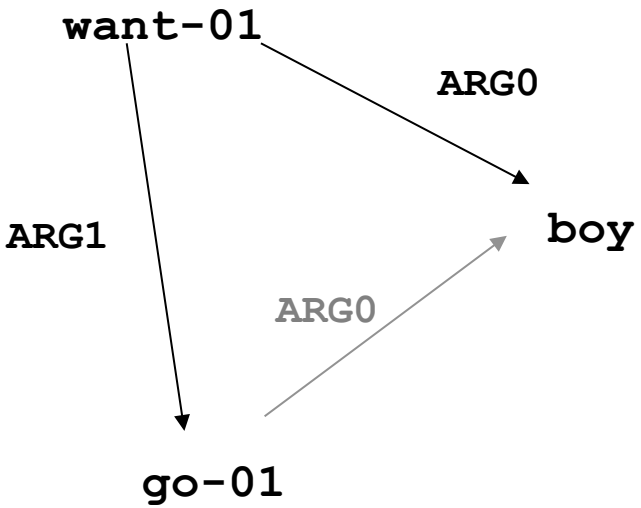


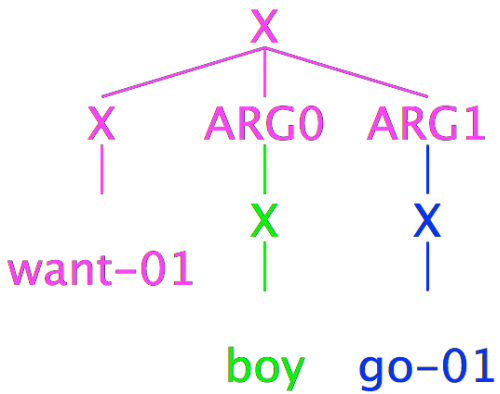
the boy wants to [2]

Rule inventory

```
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants a [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants to [2]
(X go-01) ||| go
(X go-01) ||| going
(X boy) ||| the boy
```

Decoding



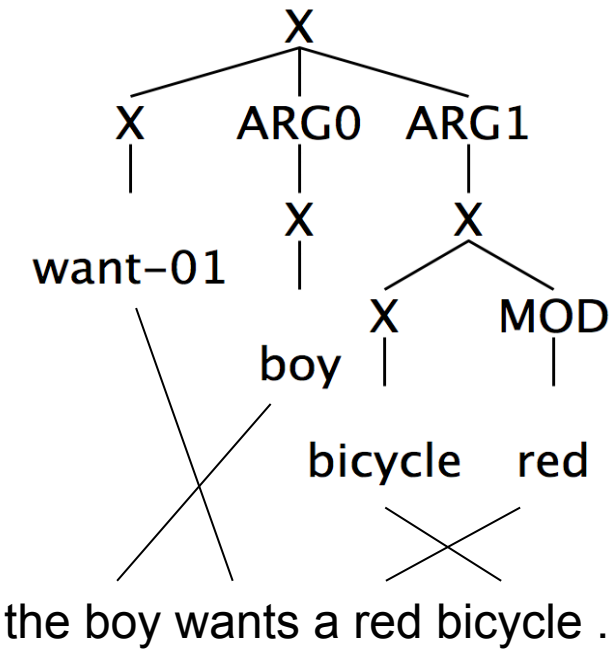


the boy wants to go

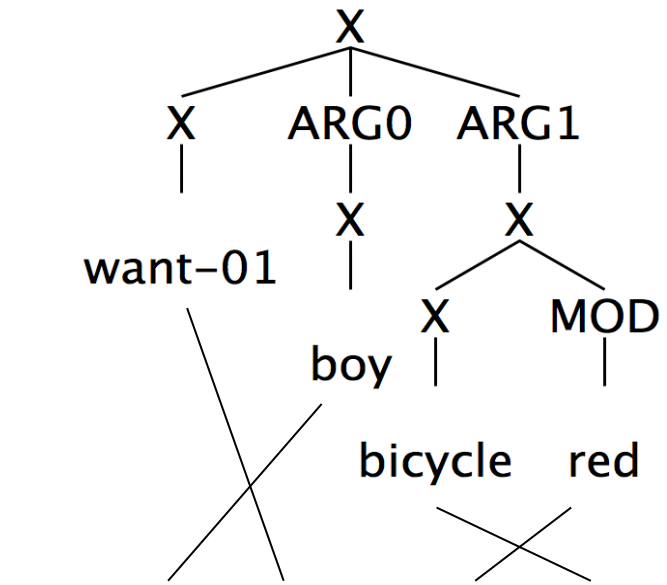
Rule inventory

```
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants a [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants to [2]
(X go-01) ||| go
(X go-01) ||| going
(X boy) ||| the boy
```

Rule Extraction



Rule Extraction




```
(X red) ||| red
(X (X bicycle [MOD])) ||| [1] bicycle
(X boy) ||| the boy
(X boy) ||| boy
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants [2]
(X (X want-01) [ARG0] [ARG1]) ||| [1] wants a [2]
```

Problem: Unseen Combinations of Arguments

```
(w / yell-01
    :ARG0 (b / (boy
        :MOD (l / little)))
```

Rule inventory

```
(X (X yell-01) [ARG0] [ARG1]) ||| [1] yelled [2]
(X (X yell-01) [ARG0] [ARG1]) ||| [1] yells [2]
(X boy) ||| the boy
(X little) ||| little
```

Synthetic Rules

```
(w / yell-01
    :ARG0 (b / (boy
        :MOD (l / little)))
```

Rule inventory

```
(X (X yell-01) [ARG0] [ARG1]) ||| [1] yelled [2]
(X (X yell-01) [ARG0] [ARG1]) ||| [1] yells [2]
(X boy) ||| the boy
(X little) ||| little
(X (X yell-01) [ARG0]) ||| [1] yelled
(X (x boy [MOD])) ||| the [1] boy
```

Synthetic Rules

```
(w / yell-01
    :ARG0 (b / (boy
        :MOD (l / little)))
```

Rule inventory

```
(X (X yell-01) [ARG0] [ARG1]) ||| [1] yelled [2]
(X (X yell-01) [ARG0] [ARG1]) ||| [1] yells [2]
(X boy) ||| the boy
(X little) ||| little
(X (X yell-01) [ARG0]) ||| [1] yelled
(X (x boy [MOD])) ||| the [1] boy
```

the little boy yelled

Synthetic Rule Model

- Associate function words with arguments

```
(X (X arrest-01) [ARG0] [ARG1] [ARG2]) ||| [1] arrested [2] for [3]
(X (X arrest-01) [ARG1] [LOC] [TIME]) ||| [3] arrests of [1] have occurred at [2]
(X (X arrest-01) [ARG1] [LOC]) ||| [1] have been arrested in [2]
```

- Use to define set of possible realizations

```
(X (X arrest-01) [LOC]) ||| arrests have occurred at [1]
(X (X arrest-01) [LOC]) ||| arrested have occurred at [1]
(X (X arrest-01) [LOC]) ||| arrests in [1]
(X (X arrest-01) [LOC]) ||| arrested in [1]
```

- Score with features:
  - Distance from core
  - Observed before (binary)
  - Etc.

Example Output (w/o synthetic rules)

```
(s / state-01
  :ARG0 (g / government-organization
        :name (n / name :op1 "Central" :op2 "Narcotics"  :op3 "Bureau")
        :poss (c / country
              :name (n2 / name
                    :op1 "Singapore")))
  :ARG1 (a / arrest-01
        :ARG1 (p2 / person
              :name (n4 / name :op1 "Michael" :op2 "Karras")
              :mod (c3 / country :name (n5 / name :op1 "Australia")))
        :ARG2 (f / find-01
              :ARG1 (p / possess-01
                    :ARG0 p2
                    :ARG1 (c4 / cannabis
                          :quant (m2 / mass-quantity
                                :quant 495
                                :unit (g4 / gram)))))
        :time (d / date-entity :day 9  :month 1 :year 2007))
  :time (d3 / date-entity :day 16 :month 1 :year 2007))
```

Output: Singaporean Bureau Narcotics Central stated on one 2007 16 Karras Michael Australia ,was arrested on one nine 2007 with 495 gram of cannabis

Ref: Singapore's Central Narcotics Bureau said on 070116 in a statement that Australian Michael Karras was arrested on 070109 after being found in possession of 495 grams of cannabis.

Future Work

- Better aligner (projectivity and co-reference)
- Regular Graph Grammars instead of tree transducers
- MT experiments (Zh->En, Cz<->En)