Abstract Meaning Representation Parsing and Generation

Speaker: Jeffrey Flanigan (CMU)

Motivation

- Goal: Translate using Abstract Meaning Representation (AMR)
- Have:
  - AMR parser
- Want:
  - AMR generator

Translation with AMR may require **less data** because it is **linguistically informed**

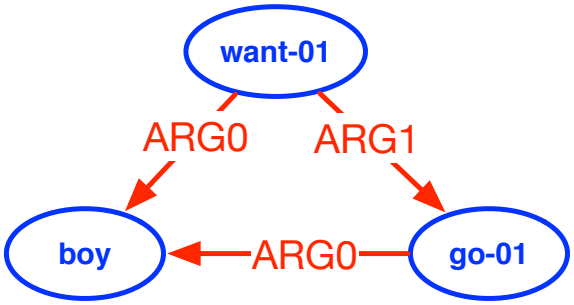
Outline

- Parsing (*ACL 2014 Honorable Mention for Best Paper*)
- Generation (*Ongoing*)
- Future Work

AMR Graphs

The boy wants to go.

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




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

Parsing into AMR

Approximately 11000 guards patrol the 1200 - kilometre border between Russia and Afghanistan.

Parsing into AMR

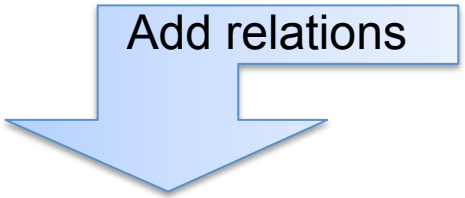
Approximately 11000 guards patrol the 1200 -
kilometre border between Russia and Afghanistan.

Invoked concepts:

```
a2 / approximately
11000
p / patrol-01
g / guard
(d4 / distance-quantity :unit (k2 / kilometer)
1200
b / border
b2 / between
(c / country :name (n / name :op1 "Russia"))
(c2 / country :name (n2 / name :op1 "Afghanistan"))
```

Parsing into AMR

Approximately 11000 guards patrol the 1200 -kilometre border between Russia and Afghanistan.




```
a2 / approximately
11000
p / patrol-01
g / guard
(d4 / distance-quantity :unit (k2 / kilometer)
1200
b / border
b2 / between
(c / country :name (n / name :op1 "Russia"))
(c2 / country :name (n2 / name :op1 "Afghanistan"))
```

```
(p / patrol-01
  :ARG0 (g / guard
          :quant (a2 / approximately
                   :op1 11000))
  :ARG1 (b / border
          :quant (d4 / distance-quantity
                   :unit (k2 / kilometer)
                   :quant 1200)
          :location (b2 / between
                      :op1 (c / country
                             :name (n / name
                                      :op1 "Russia"))
                      :op2 (c2 / country
                             :name (n2 / name
                                      :op1 "Afghanistan"))))))
```

Parsing into AMR

Approximately 11000 guards patrol the 1200 -kilometre border between Russia and Afghanistan.

Sequence labeling task
well-studied algorithms exist

```
a2 / approximately
11000
p / patrol-01
g / guard
(d4 / distance-quantity :unit (k2 / kilometer)
1200
b / border
b2 / between
(c / country :name (n / name :op1 "Russia"))
(c2 / country :name (n2 / name :op1 "Afghanistan"))
```

```
(p / patrol-01
  :ARG0 (g / guard
          :quant (a2 / approximately
                   :op1 11000))
  :ARG1 (b / border
          :quant (d4 / distance-quantity
                   :unit (k2 / kilometer)
                   :quant 1200)
          :location (b2 / between
                      :op1 (c / country
                             :name (n / name
                                      :op1 "Russia"))
                      :op2 (c2 / country
                             :name (n2 / name
                                      :op1 "Afghanistan")))))
```

Find the **maximum-weighted, connected, spanning graph**
(with additional constraints)

Technical Details

- Edge factored: weight for each relation in isolation
  - Computationally attractive
  - Most semantic parsers assume this
- Assume rich input:
  - Output of syntactic parsers, POS taggers, semantic features as input

Performance

- Train on 4k annotated sentences (low resource scenario)
- Test on 2k annotated sentences
- Evaluated using Smatch (Cai & Knight, 2013)

Precision:	65%
Recall:	57%
F1:	61%

Output

France seeks civilian nuclear energy cooperation with Arab States in order to isolate Iran over nuclear standoff.

Output AMR

```
(c / country, name, n2 / name)
(c2 / country, name, n3 / name)
(c3 / cooperate-01, ARG0, c2 / country)
(c3 / cooperate-01, ARG2, e / energy)
(c3 / cooperate-01, mod, c4 / civilian)
(c5 / country, name, n5 / name)
(e / energy, mod, n4 / nucleus)

(i / isolate-01, ARG0, c5 / country)
(i / isolate-01, ARG1, c / country)
(i / isolate-01, location, o / over)
(n2 / name, op, "Iran")
(n3 / name, op, "Arab")
(n3 / name, op, "States")
(n5 / name, op, "France")
(o / over, op, n / nucleus)
(o / over, op, s / standoff)
(o2 / order-02, ARG1, i / isolate-01)
(s2 / seek-01, ARG0, c5 / country)
(s2 / seek-01, ARG1, c3 / cooperate-01)
```

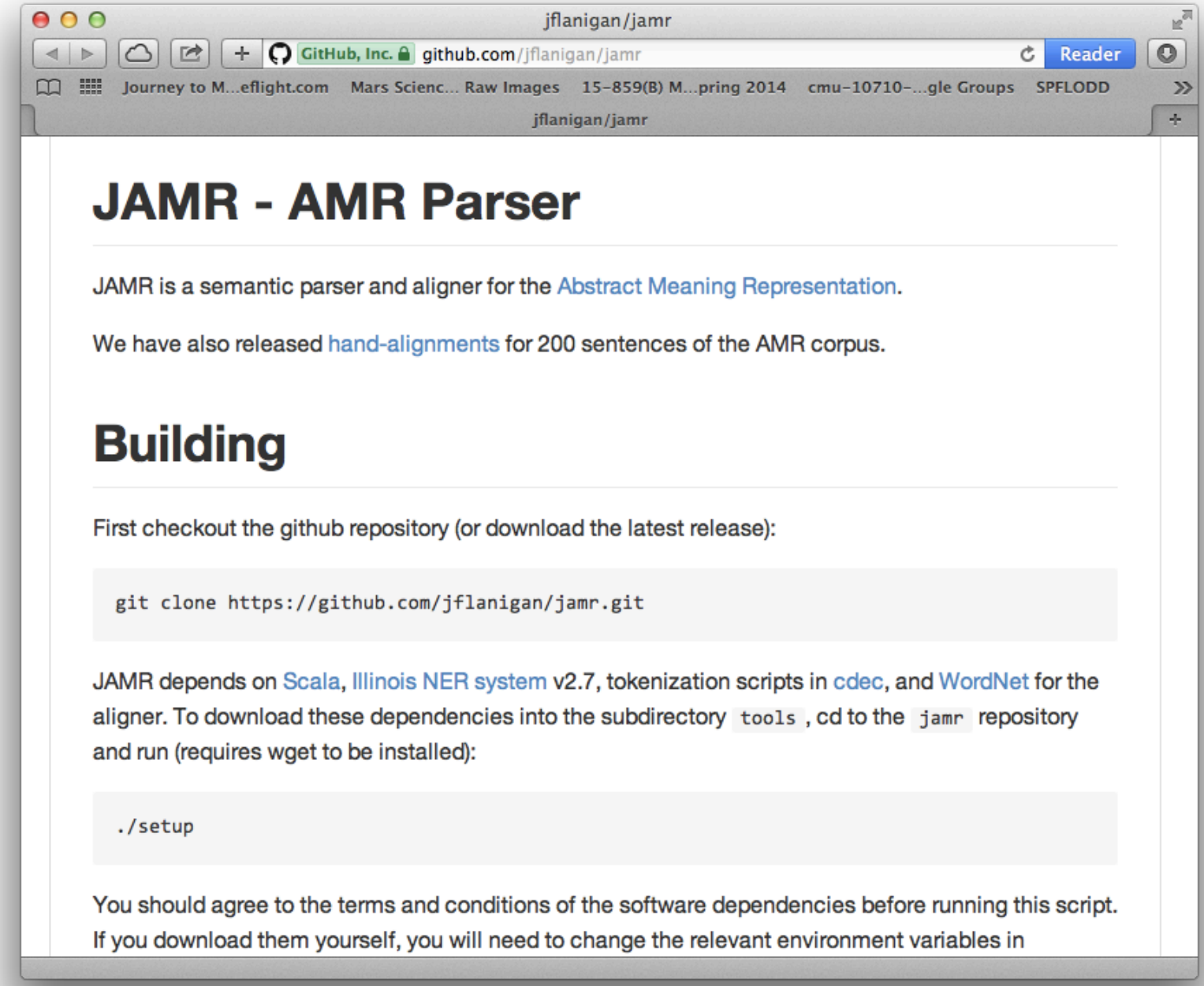
Human annotated AMR (differences):

```
(e2/ethnic-group, name, n2/name)

(c2/cooperate-01, purpose, i/isolate-01)

(e/energy, mod, c3/civilian)

(s2 / standoff, mod, n3 / nucleus)
```



JAMR - AMR Parser

JAMR is a semantic parser and aligner for the Abstract Meaning Representation.

We have also released hand-alignments for 200 sentences of the AMR corpus.

Building

First checkout the github repository (or download the latest release):

```
git clone https://github.com/jflanigan/jamr.git
```

JAMR depends on Scala, Illinois NER system v2.7, tokenization scripts in cdec, and WordNet for the aligner. To download these dependencies into the subdirectory `tools`, cd to the `jamr` repository and run (requires wget to be installed):

```
./setup
```

You should agree to the terms and conditions of the software dependencies before running this script. If you download them yourself, you will need to change the relevant environment variables in

Generation

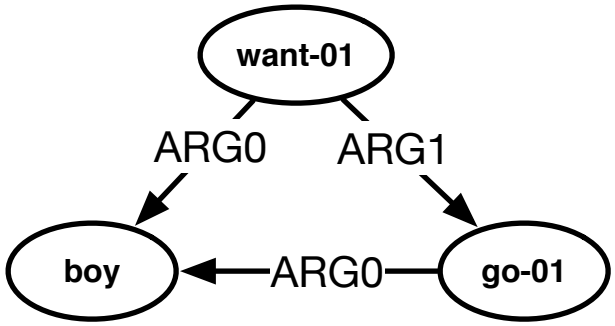
```
(p / patrol-01
  :ARG0 (g / guard
          :quant (a2 / approximately
                   :op1 11000))
  :ARG1 (b / border
          :quant (d4 / distance-quantity
                   :unit (k2 / kilometer)
                   :quant 1200)
          :location (b2 / between
                      :op1 (c / country
                             :name (n / name
                                      :op1 "Russia"))
                      :op2 (c2 / country
                             :name (n2 / name
                                      :op1 "Afghanistan")))))
```

Approximately 11000 guards patrol the 1200 - kilometre border between Russia and Afghanistan.

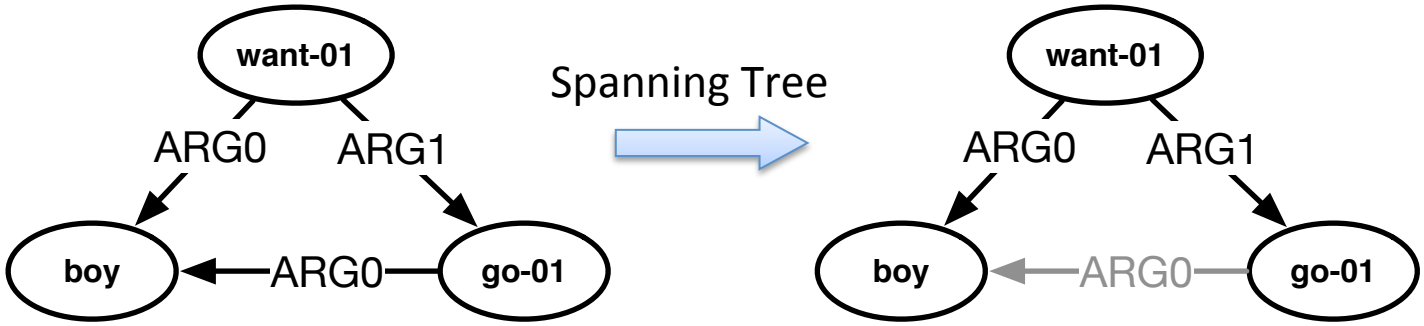
Generation

- Compute spanning tree(s)
  - Output will be projective in this tree
- Transducer rules define output space
  - Rules learned from corpus
  - Also create synthetic rules on-the-fly
- Search for best output
  - Language model
  - Other features as well

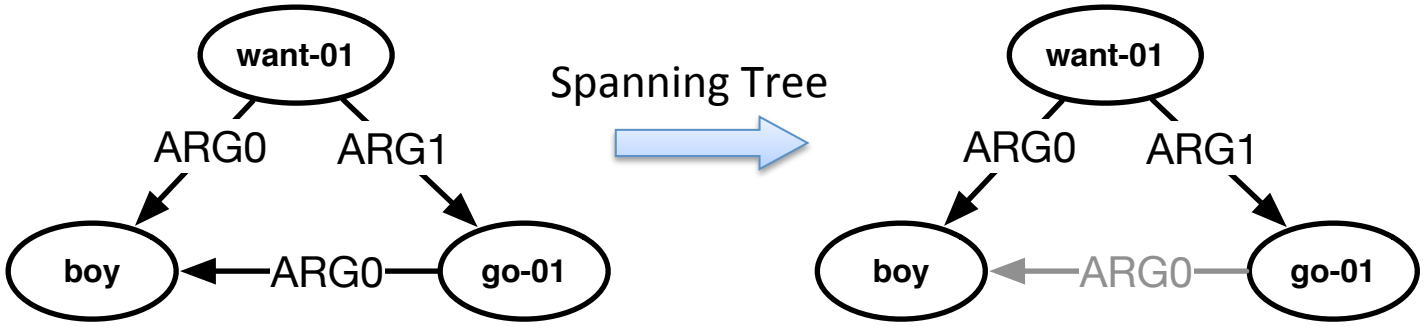
Generation



Generation


want-01
ARG0
ARG1
boy
ARG0
go-01
Spanning Tree
want-01
ARG0
ARG1
boy
ARG0
go-01

Generation



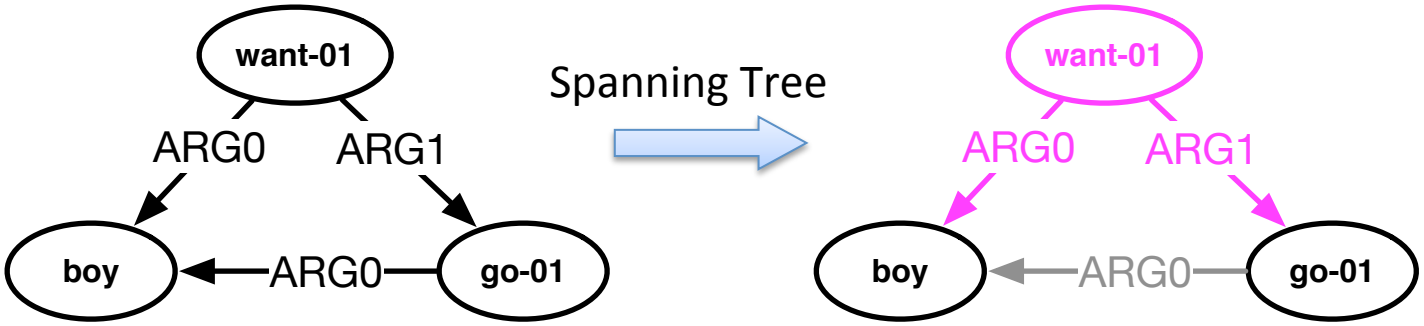
Concept and relations

English realization

Rule inventory

```
want-01 [ARG0] [ARG1] ||| [ARG0] wants [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants a [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants to [ARG1]
go-01 ||| go
go-01 ||| going
boy ||| the boy
```

Generation

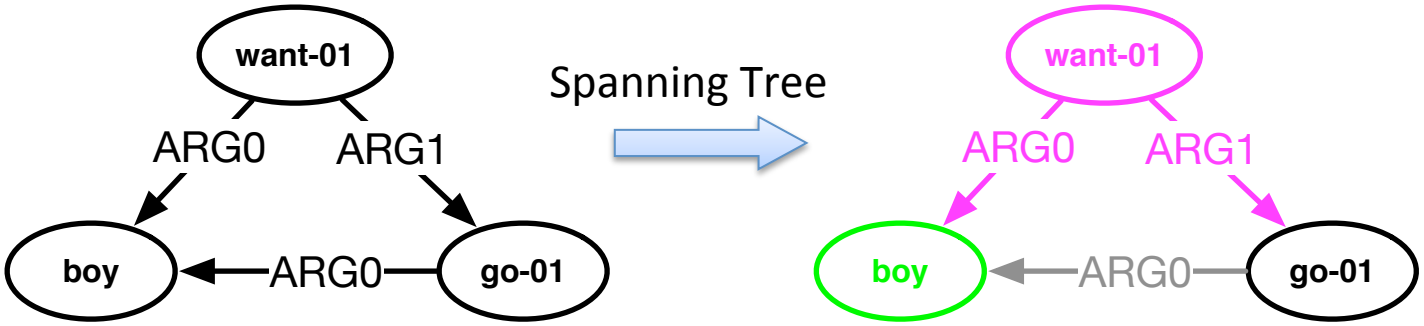


[ARG0] wants to [ARG1]

Rule inventory

```
want-01 [ARG0] [ARG1] ||| [ARG0] wants [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants a [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants to [ARG1]
go-01 ||| go
go-01 ||| going
boy ||| the boy
```

Generation

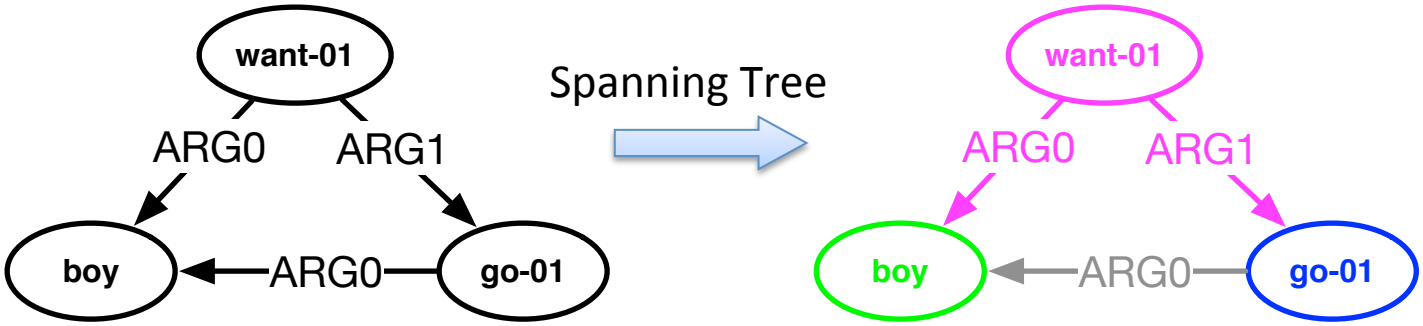


the boy wants to [ARG1]

Rule inventory

```
want-01 [ARG0] [ARG1] ||| [ARG0] wants [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants a [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants to [ARG1]
go-01 ||| go
go-01 ||| going
boy ||| the boy
```

Generation

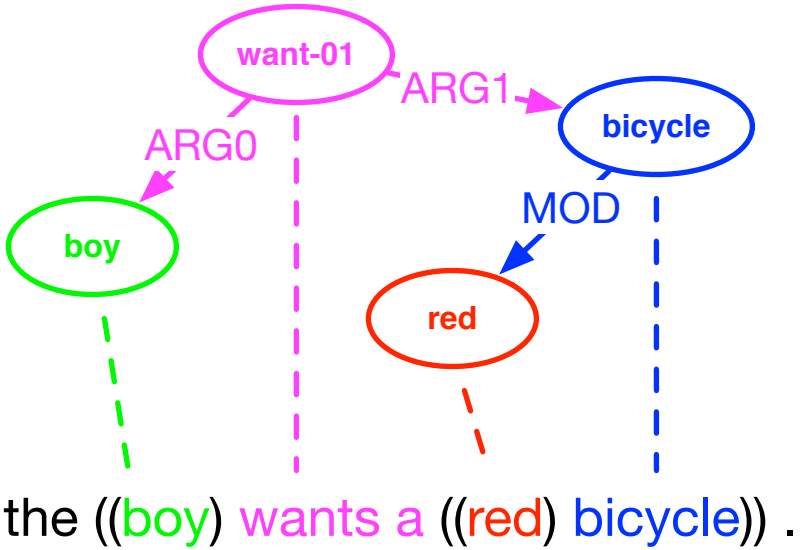


the boy wants to go

Rule inventory

```
want-01 [ARG0] [ARG1] ||| [ARG0] wants [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants a [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants to [ARG1]
go-01 ||| go
go-01 ||| going
boy ||| the boy
```

Rule Extraction




```
red ||| red
bicycle [MOD] ||| [MOD] bicycle
boy ||| boy
want-01 [ARG0] [ARG1] ||| [ARG0] wants [ARG1]
want-01 [ARG0] [ARG1] ||| [ARG0] wants a [ARG1]
```

Problem

Rule inventory:

```
yell-01 [ARG0] [ARG1] ||| [ARG0] yelled [ARG1]
yell-01 [ARG0] [ARG1] ||| [ARG0] yells [ARG1]
boy ||| the boy
little ||| little
```

Input AMR:

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

No rule for:

```
yell-01 [ARG0] ||| ???
boy [MOD] ||| ???
```

Solution: Synthetic Rules

Input AMR:

```
(a / arrest-01
   :LOC airport)
```

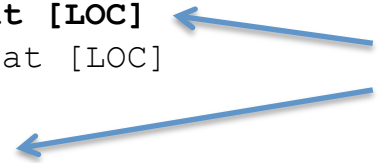
- Rules extracted from corpus:

```
arrest-01 [ARG0] [ARG1] [ARG2] ||| [ARG0] arrested [ARG1] for [ARG2]
arrest-01 [ARG1] [LOC] [TIME] ||| [TIME] arrests of [ARG1] occurred at [LOC]
arrest-01 [ARG1] [LOC] ||| [ARG1] have been arrested in [LOC]
airport ||| the airport
```

Need rule for: **`arrest-01 [LOC] ||| ???`**

- Possible realizations:

```
arrest-01 [LOC] ||| arrests occurred at [LOC]
arrest-01 [LOC] ||| arrested occurred at [LOC]
arrest-01 [LOC] ||| arrests in [LOC]
arrest-01 [LOC] ||| arrested in [LOC]
```



Synthetic rules

Output: **`arrests occurred at the airport`**

Example Output (w/o synthetic rules)

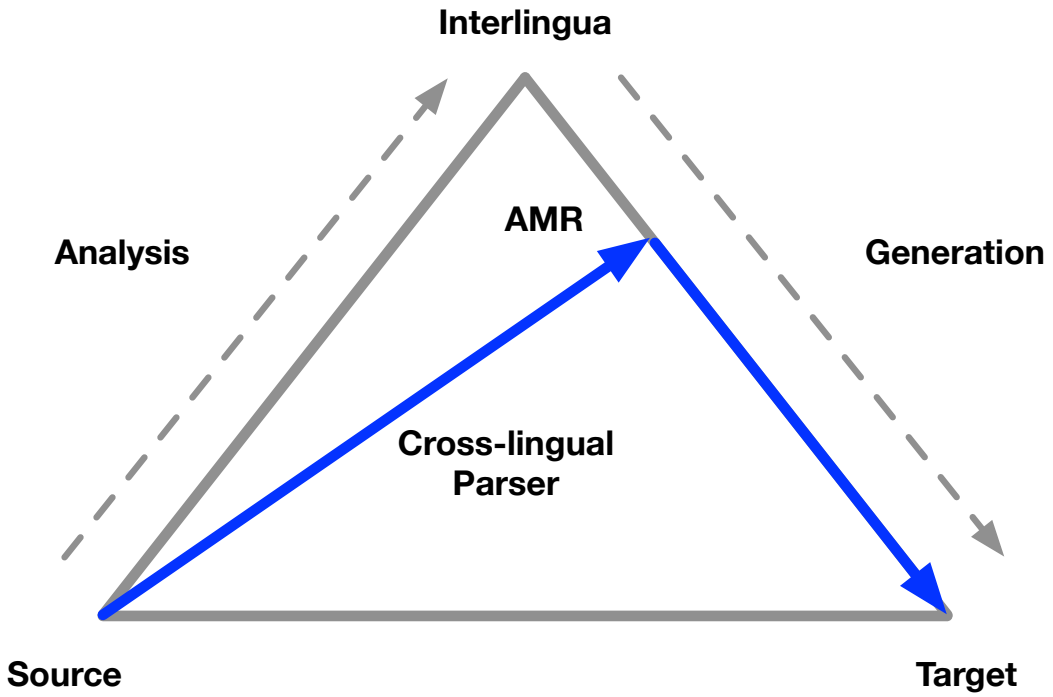
```
(s / state-01
  :ARG0 (g / government-organization
        :name (n / name :op1 "Central" :op2 "Narcotics"  :op3 "Bureau")
        :poss (c / country
              :name (n2 / name
                    :op1 "Singapore")))
  :ARG1 (a / arrest-01
        :ARG1 (p2 / person
              :name (n4 / 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

- Cross lingual AMR parsing
  - Run AMR parser on bilingual corpus
  - Train cross-lingual AMR parser



Thank you!

```
(t / thank-01
     :ARG1 (y / you))
```

Backup Slides

Rule Extraction

```
1 (X (X arrest-01) [ARG0] [ARG1] [ARG2]) ||| [1] arrested [2] for [3]
1 (X (X arrest-01) [ARG0] [ARG1] [MANNER]) ||| The [1]  [3] arrested [2]
2 (X (X arrest-01) [ARG0] [ARG1] [TIME]) ||| [3]  [1] arrested [2]
1 (X (X arrest-01) [ARG0] [QUANT] [TIME]) ||| [1] made [2] arrests [3]
1 (X (X arrest-01) [ARG1] [ACCOMPANIER] [TIME]) ||| [1] was arrested with [2] in [3]
1 (X (X arrest-01) [ARG1] [LOCATION] [TIME]) ||| [1] is under [2] arrest [3]
1 (X (X arrest-01) [ARG1] [LOCATION] [TIME]) ||| [1] was arrested in [2] in [3]
1 (X (X arrest-01) [ARG1] [LOCATION] [TIME]) ||| [3] arrests of [1] have occurred at [2]
1 (X (X arrest-01) [ARG1] [LOCATION]) ||| [1] have been arrested in [2]
1 (X (X arrest-01) [ARG1] [TIME]) ||| [1] was arrested [2]
1 (X (X arrest-01) [ARG1] [TIME]) ||| [1] were arrested in [2]
1 (X (X arrest-01) [ARG1]) ||| [1] were arrested
```

Argument Realization Patterns

⇒ long tail

```
1478 [MOD] NN
 496 [ARG1] NN
 373 VB [ARG1]
 269 [ARG0] VBD that [ARG1]
 248 [ARG0] VBD [ARG1]
 232 VBG [ARG1]
 222 NN of [ARG1]
 123 [MOD] [MOD] NN
 111 [ARG1OF] NN
 102 [ARG0] NN
 102 JJ [ARG1]
 100 [QUANT] NN
  79 NN in [LOCATION]
  79 NN [ARG1]
  73 [NAME] NN
  66 [MOD] JJ
  66 NN to [ARG1]
  65 [ARG2] NN
  65 [ARG0OF] NN
  64 [LOCATION] NN
  59 NN [ARG1OF]
  52 NN of [POSS]
  49 [DEGREE] JJ
  49 IN [ARG2]
  46 [POSS] NN
  44 [ARG0] VBP [ARG1]
...
   1 [ARG0] VBG [ARG1] [TIME]
   1 [ARG0] VBD with [ARG2] to [ARG1]
   1 [ARG0] VBD with [ARG2] [MANNR]
   1 [ARG0] VBD with [ARG2]
   1 [ARG0] VBD with [ARG1]
   1 [ARG0] VBD to [PURPOSE]
   1 [ARG0] VBD to [DESTINATION] [TIME]
   1 [ARG0] VBD to [DESTINATION]
   1 [ARG0] VBD to [ARG2] on [TIME] that [ARG1]
   1 [ARG0] VBD to [ARG1] with [ARG2]
   1 [ARG0] VBD to [ARG1] [ARG1OF]
   1 [ARG0] VBD their [ARG1] with [MANNR]
   1 [ARG0] VBD their [ARG1] [TIME]
```

Lexically dependent

```
 5 NN to [ARG1] ||| visit-01
 1 [ARG0] 's NN ||| visit-01
 1 NN by [ARG0] ||| visit-01

 5 NN of [ARG1] ||| construct-01
 2 [ARG1] NN ||| construct-01
 2 NN on [ARG1] ||| construct-01
 1 [ARG1] 's NN ||| construct-01

 6 [ARG1] NN ||| cultivate-01
 5 NN of [ARG1] ||| cultivate-01

 4 NN to [ARG1] ||| access-01
 1 [ARG1] NN ||| access-01
 1 [ARG0] NN ||| access-01

15 [ARG1] NN ||| smuggle-01
 1 NN from [ARG2] ||| smuggle-01

 5 [ARG1] NN ||| inspect-01
 1 [ARG0] NN ||| inspect-01
 1 NN of [ARG1] ||| inspect-01
 1 NN at [ARG1] ||| inspect-01

 5 [ARG1] NN ||| exchange-01
 4 NN of [ARG1] ||| exchange-01
 1 NN with [ARG0] ||| exchange-01
 1 NN [ARG1] ||| exchange-01

 8 [ARG0] NN ||| attack-01
 5 [ARG1] NN ||| attack-01
 4 NN by [ARG0] ||| attack-01
 1 NN on [ARG1] ||| attack-01
 1 NN against [ARG1] ||| attack-01
 1 NN [ARG1] ||| attack-01
```

Solution: Synthetic Rules

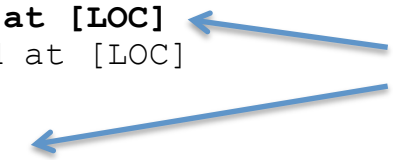
Need rule for: **`arrest-01 [LOC]`**

- Rules extracted from corpus:

```
arrest-01 [ARG0] [ARG1] [ARG2] ||| [ARG0] arrested [ARG1] for [ARG2]
arrest-01 [ARG1] [LOC] [TIME] ||| [TIME] arrests of [ARG1] have occurred at [LOC]
arrest-01 [ARG1] [LOC] ||| [ARG1] have been arrested in [LOC]
```

- Possible realizations:

```
arrest-01 [LOC] ||| arrests have occurred at [LOC]
arrest-01 [LOC] ||| arrested have occurred at [LOC]
arrest-01 [LOC] ||| arrests in [LOC]
arrest-01 [LOC] ||| arrested in [LOC]
```



Add as synthetic rules

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