NLP Chunking for Game Rules Parsing

MS548 NLP Teach-back

Jake O'Connor

What is Chunking

- Extracting meaning from patterns in natural language.
- Heavily relies on POS tagging.
- Patterns depend on language and context.
- Chunks can range from general or specific (chunking-up or -down).
- Often visualized as trees.

What are Game Rules

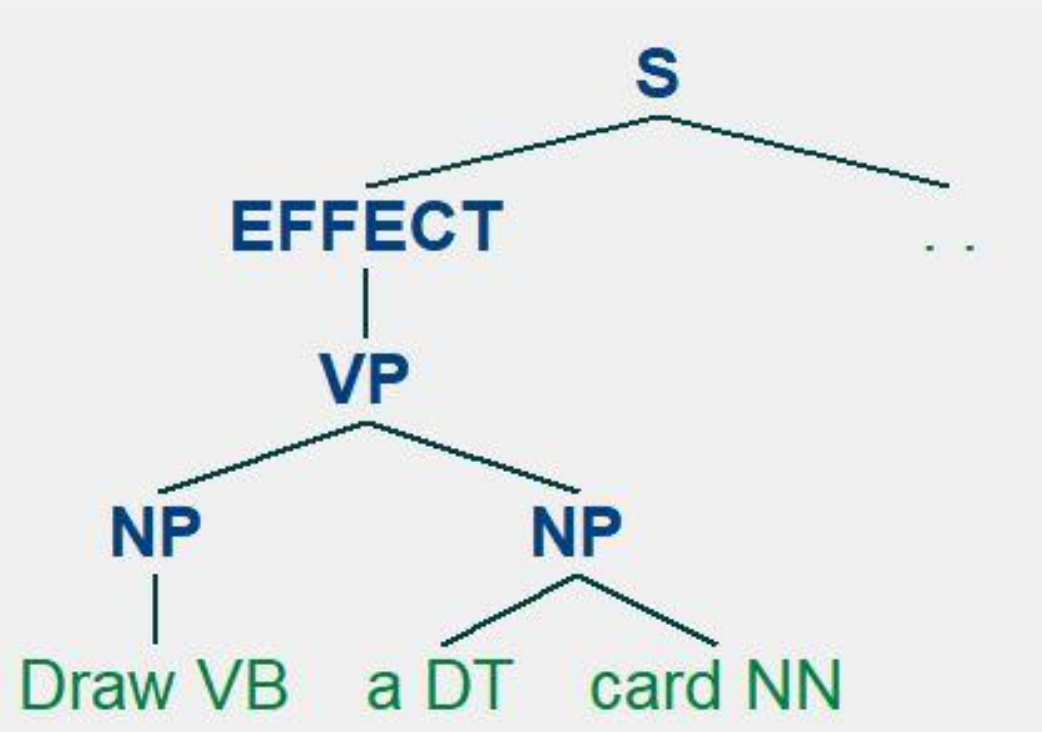
Natural language describing game logic.

Written by non-engineers using human-readable text.

Specific words and phrases have functional meanings.

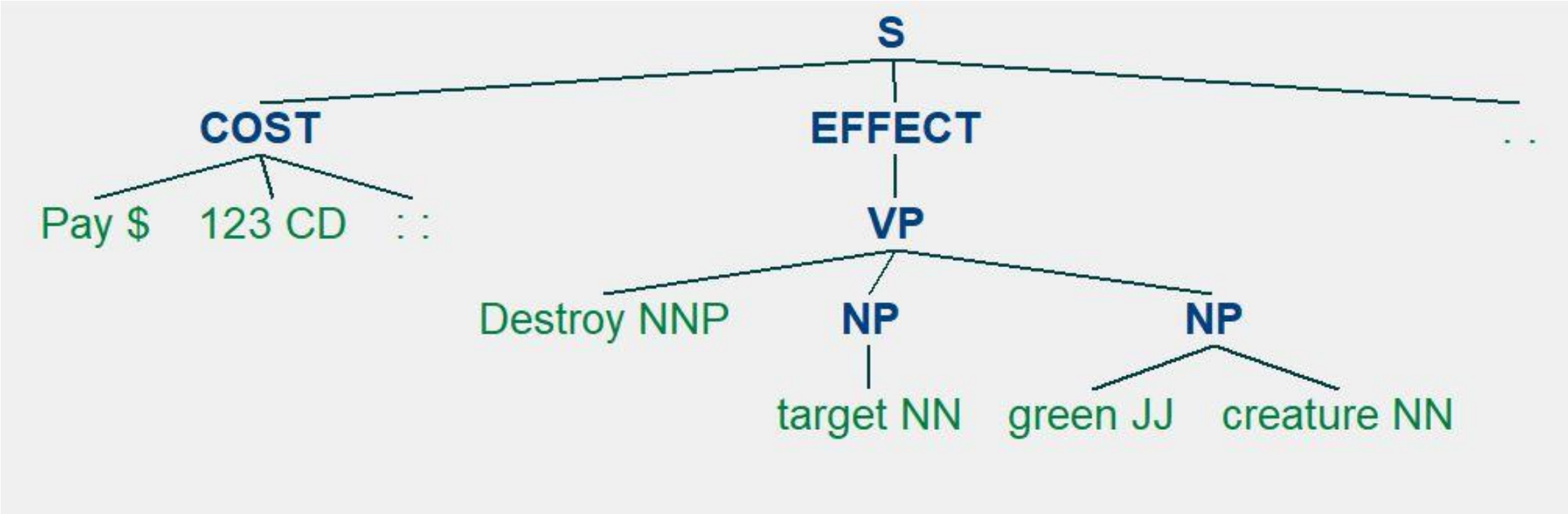
"Draw a card."

- Very simple example.
- Single effect, with no cost or additional parts.



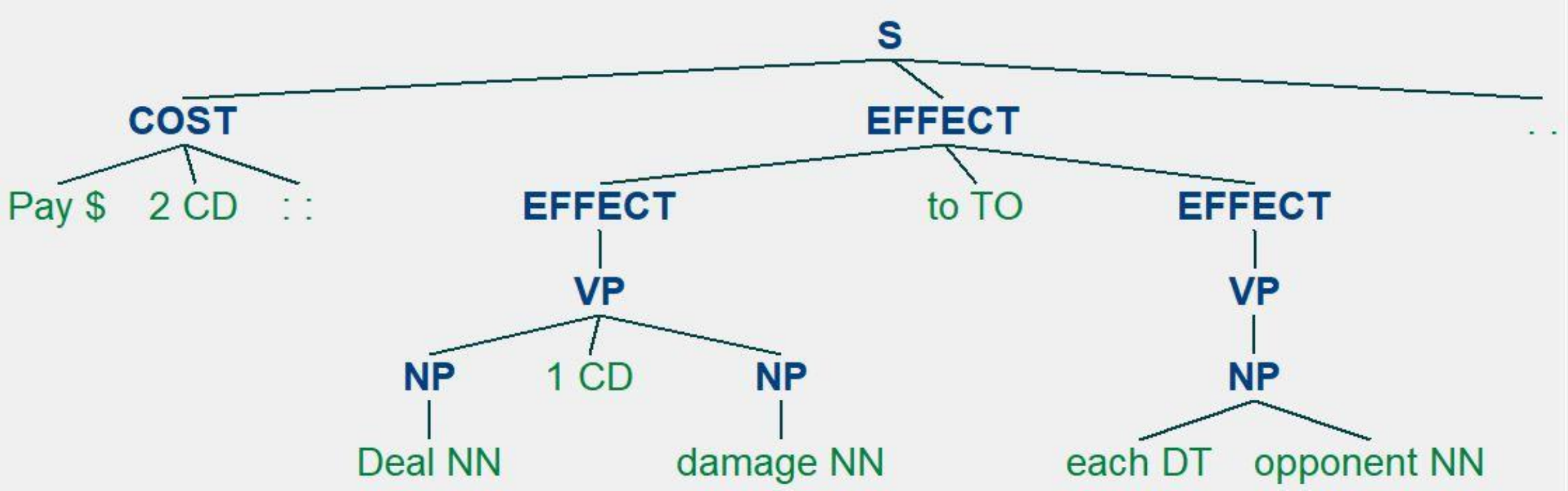
"Pay 123: Destroy target green creature."

- Intermediate example.
- Includes both a cost and effect group, including noun qualifiers.



"Pay 2: Deal 1 damage to each opponent."

- Advanced example.
- Cost phrase, as well as a complex Effect phrase with numbers, determiners, and an action target.



How does any of this work?

```python
# chunker which is designed to parse (relatively simple) ability text into meaningful
chunker = RegexpParser(
    '''
    NP: {<DT|VB>? <JJ>* <NN>*}
    V: {<V.*>}
    VP: {<V|NP|NNP> <CD>? <NP|PP>*}
    COST: {<.*> <CD> <:>}
    COST: {<VP> <:>}
    EFFECT: {<V|VP> <TO> <NP>}
    EFFECT: {<VP>}
    EFFECT: {<NNP> <EFFECT>}
    EFFECT: {<EFFECT> <TO> <EFFECT>}
    ''')


def parse_ability_text(text: str) -> nltk.tree.Tree:
    tokens = nltk.word_tokenize(text)  # tokenize the string into individual words
    tags = nltk.pos_tag(tokens)  # tag each word with it's part-of-speech
    return chunker.parse(tags)  # parse tagged text into meaningful chunks tree
```

1. Tokenize the phrase into words.
2. Tag each word with its part-of-speech.
   - Default NLTK tagger used here, real code would need a better POS tagger.
3. Parse the tag collection using the Regex parser.
   - Cost defined as verb pairs or numerals followed by a colon.
   - Effect defined as verb pairs, noun pairs, and/or groups of effects.

But why would this ever be useful?

- Hard-coding logic is expensive, bad, and fragile.
- As new chunk-patterns emerge, they can be codified into game logic.
- Allows non-engineers to create, iterate, and expand upon game rules without engineer support (to an extent).
- Allows smaller teams to support larger projects.