A Controlled Natural Language Interface for Electronic Contracts

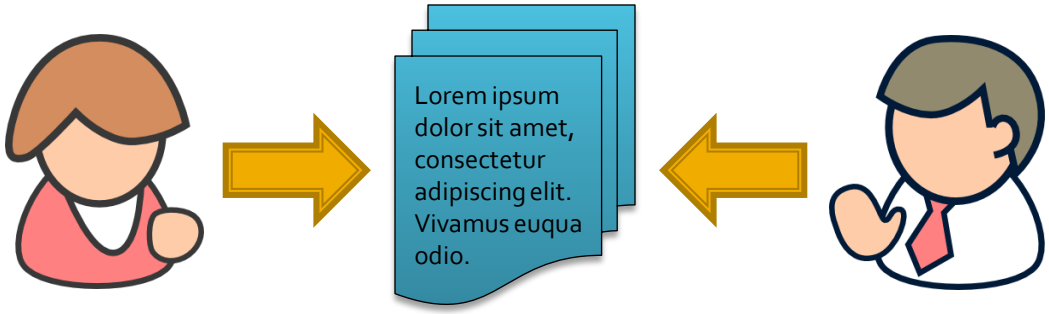
Final Year Project Presentation

John J. Camilleri
B.Sc. IT (Hons.) 4th Year

Supervised by Michael Rosner and Gordon J. Pace

What is a contract?

- Agreement between parties regulating their behaviour



- *Deontic* notions
  - Duty / obligation
  - Right / permission
  - Illegality / prohibition

Contract examples

- ISP service-level agreement
  - *Minimum download speed should be above 20kb/s*
- Laws, treaties, constitutions
  - *CO_2 emissions must not exceed 130 g/km² per year*
- Game rules
  - *As a player passes GO, they may collect £200*

Why electronic contracts?

- Contracts are written using natural language
  - Fraught with ambiguity
  - Different interpretations
  - Require paid professionals to write & check
- Formally representing contracts
  - Eliminate ambiguity
  - Real-time checking
  - Automated conflict detection

Language problem

- Example
  - *All students must submit their assignment at some point before 12:00 noon*
  - $\forall$ s : Student ·
    $\Diamond_{[0,1200]}$ O (s, submit) $\wedge$ $\Box_{[1201,\infty]}$ F (s, submit)

- Not human-friendly!
- Need to reconcile natural and formal representations

Controlled natural languages (CNLs)

- Reduced syntax / vocabulary
- Goal: *adequate* expressivity
- Benefit: a lot easier to parse / generate
  - **Natural**
    The system is forbidden from producing a result if it has been cancelled by the owner.
  - **Controlled**
    If owner of Job cancels Job, it is forbidden that SYSTEM produces result of Job
  - Pace, G. J., & Rosner, M. (2009). A Controlled Language for the Specification of Contracts. In *Workshop on Controlled Natural Language 2009 (CNL'09)*. Marettimo, Italy.

Layered approach

- Formal contract logic *underneath*
- CNL interface *on top*

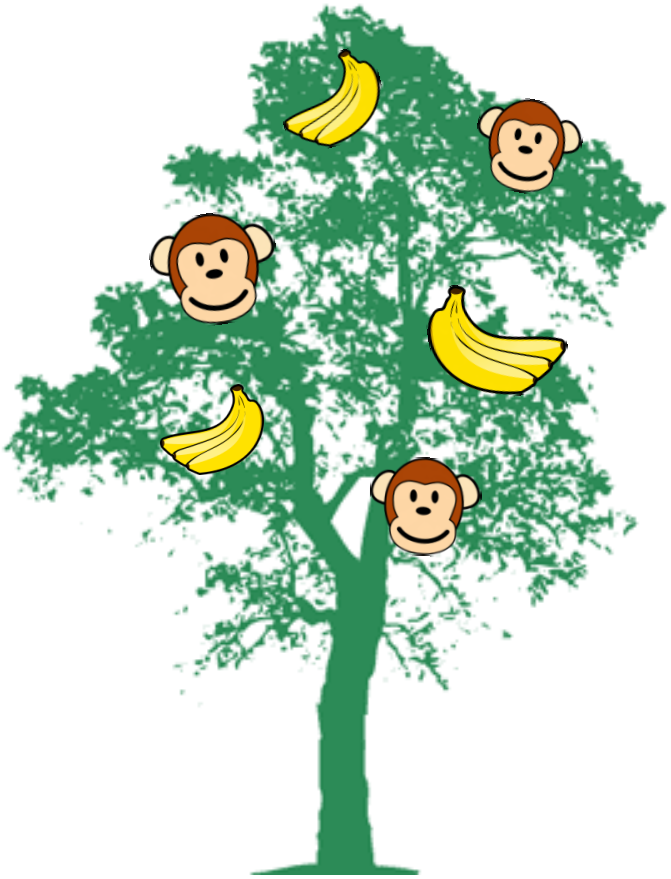
- Note: CNL is itself formal, but added abstraction is still advantageous

Case study: Nomic

- "A game of self-amendment"
  - Suber, P. (1990). Nomic: A Game of Self-Amendment. In *The Paradox of Self-Amendment*. Peter Lang Publishing. Retrieved from http://www.earlham.edu/~peters/nomic.htm.
- Turns are made by changing the rules
- Everything can change…
  - …including how you win!
- Many variations exist, all managed 'manually'
  - How to automate Nomic?

BanaNomic

- Reduced version of Nomic
- Monkeys in a tree, collecting bananas
- Basic actions
  - Climb up / down
  - Pick / throw bananas
- Rules of the rainforest



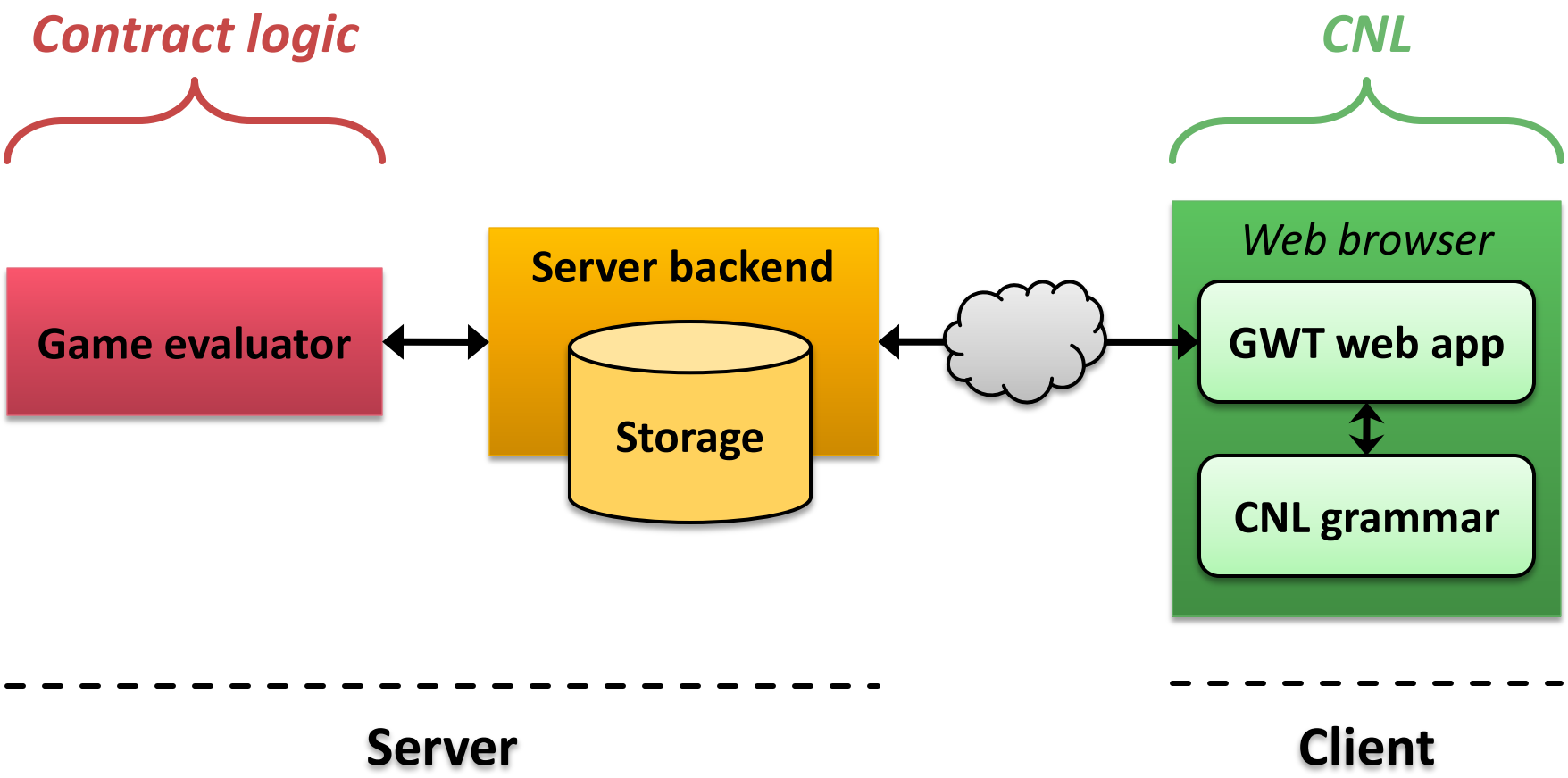
BanaNomic

- Rules of the rainforest
  - Obligations, permissions, prohibitions
- Example
  - *If Paul has more than 5 bananas then Paul is permitted to climb up the tree*
- Must be obeyed…but can be changed!
- Players can enact / abolish rules with every turn

Project objectives

1. Deontic contract logic for BanaNomic
2. Evaluator for automatic turn-checking
3. CNL interface
4. Playable implementation of game
5. Evaluate qualitatively

Architecture



Contract logic

- Haskell embedded grammar
- Declaratively defined

Formal notation

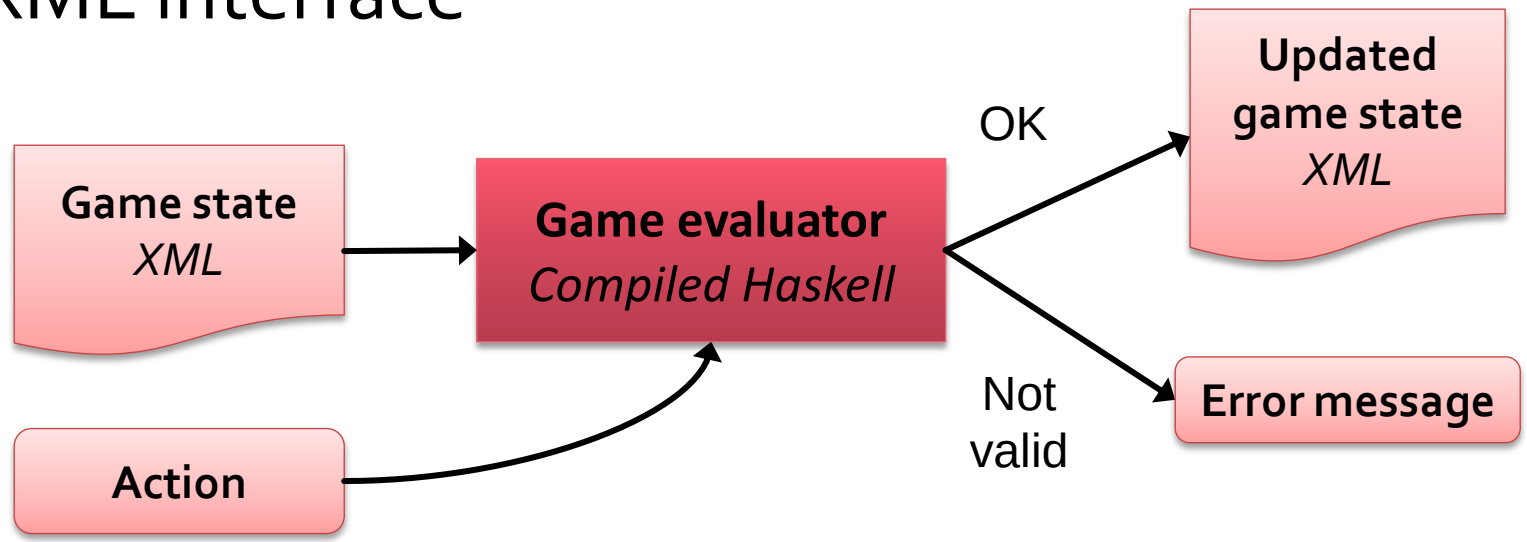
$$
\begin{aligned}
Clause \quad ::= \quad & Ok \\
\mid \quad & DeonticExp \\
\mid \quad & Clause + Clause \\
\mid \quad & \Box\,[Time, Time]\; Clause \\
\mid \quad & \Diamond\,[Time, Time]\; Clause \\
\mid \quad & Clause \lhd\!\lhd Query \rhd\!\rhd Clause \\
\mid \quad & Clause \lhd DeonticExp \rhd Clause
\end{aligned}
$$

Haskell syntax

```
data Clause =
    C_Ok
  | C_Deontic DeonticExp
  | C_Choice Clause Clause
  | C_Always Time Time Clause
  | C_Sometimes Time Time Clause
  | C_Query Query Clause Clause
  | C_Conditional DeonticExp
        Clause Clause
```

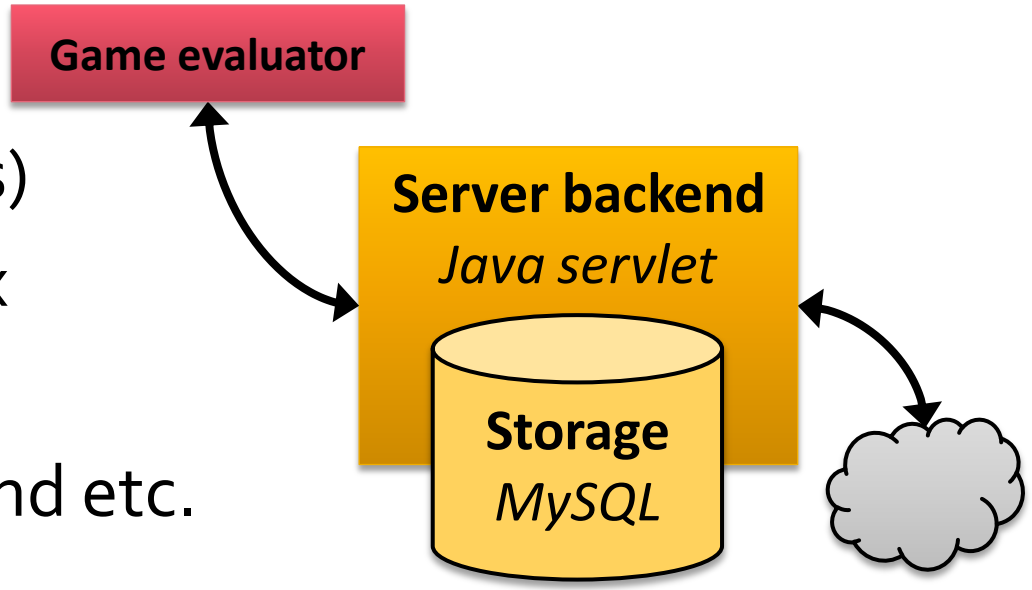
Game evaluator

- Stateless
- Validates action against contract
- Updates & returns game state if accepted
- XML interface



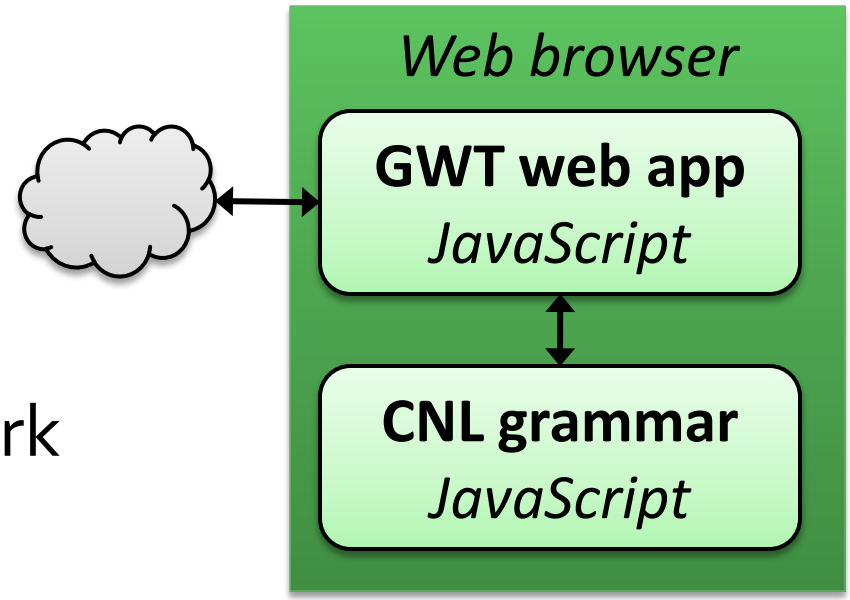
Server backend

- Storage of game states
- Conversions
  - XML (evaluator)
  - Java objects (POJOs)
  - CNL abstract syntax
- Usual server stuff
  - Logins, AJAX backend etc.



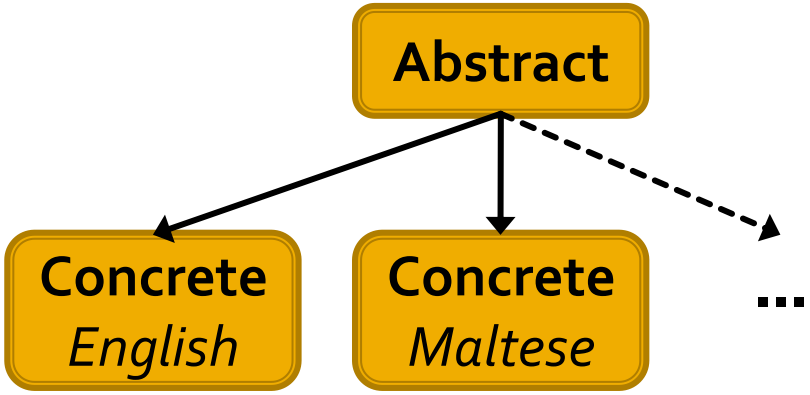
Web application

- User interface
  - Google Web Toolkit
- CNL layer
  - Grammatical Framework compiled to JavaScript
  - Phrase generation
  - Guided input methods



Grammatical Framework (GF)

- Functional language framework
- Multilingual applications
- 2-level structure
  - Abstract
  - Concrete



- Libraries for 14 languages, and growing
  - Ranta, A. (2009). The GF Resource Grammar Library. *Linguistic Issues in Language Technology*, 2(2).

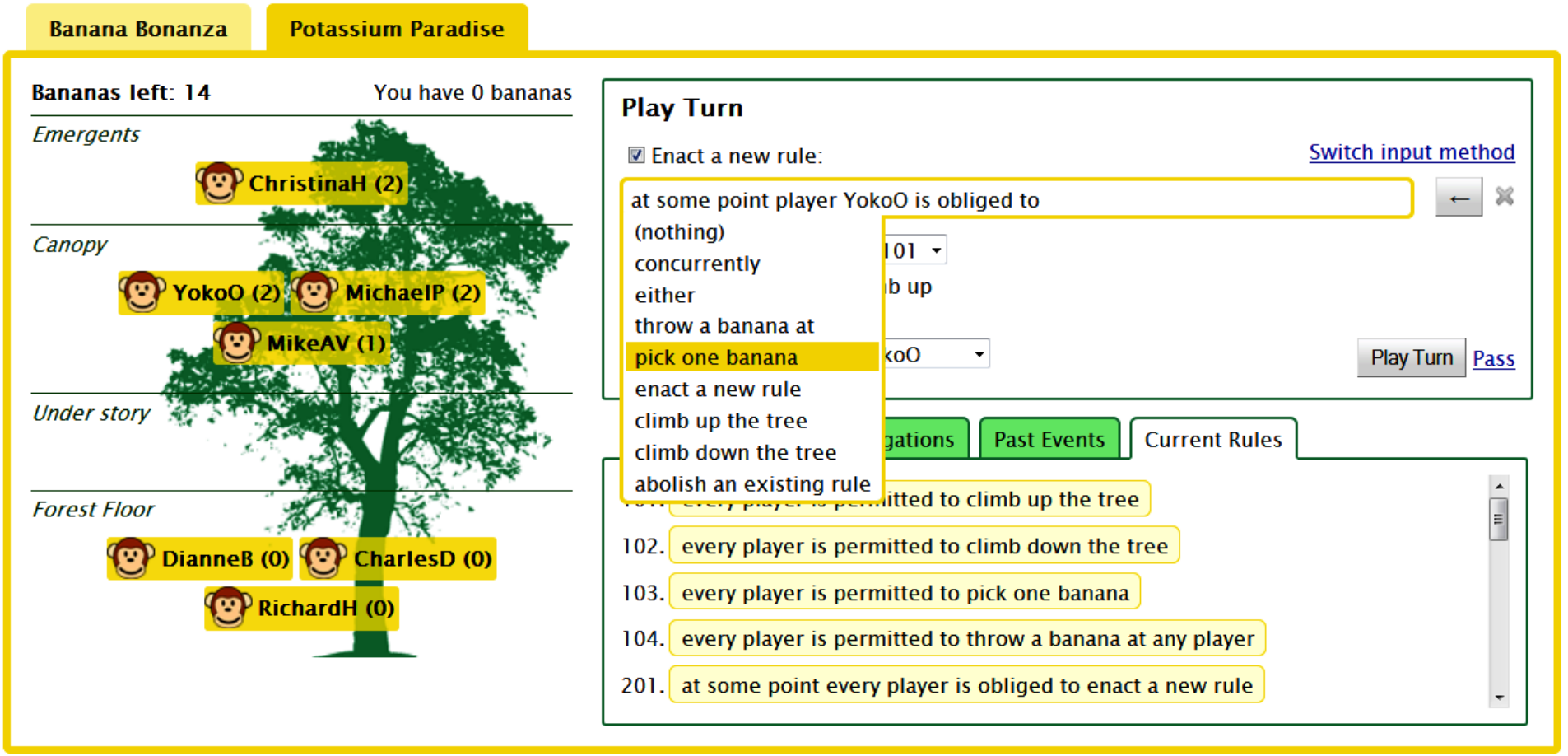
Guided input methods

- Allow *only* grammatical phrases
- "Suggest panel" (auto-complete)

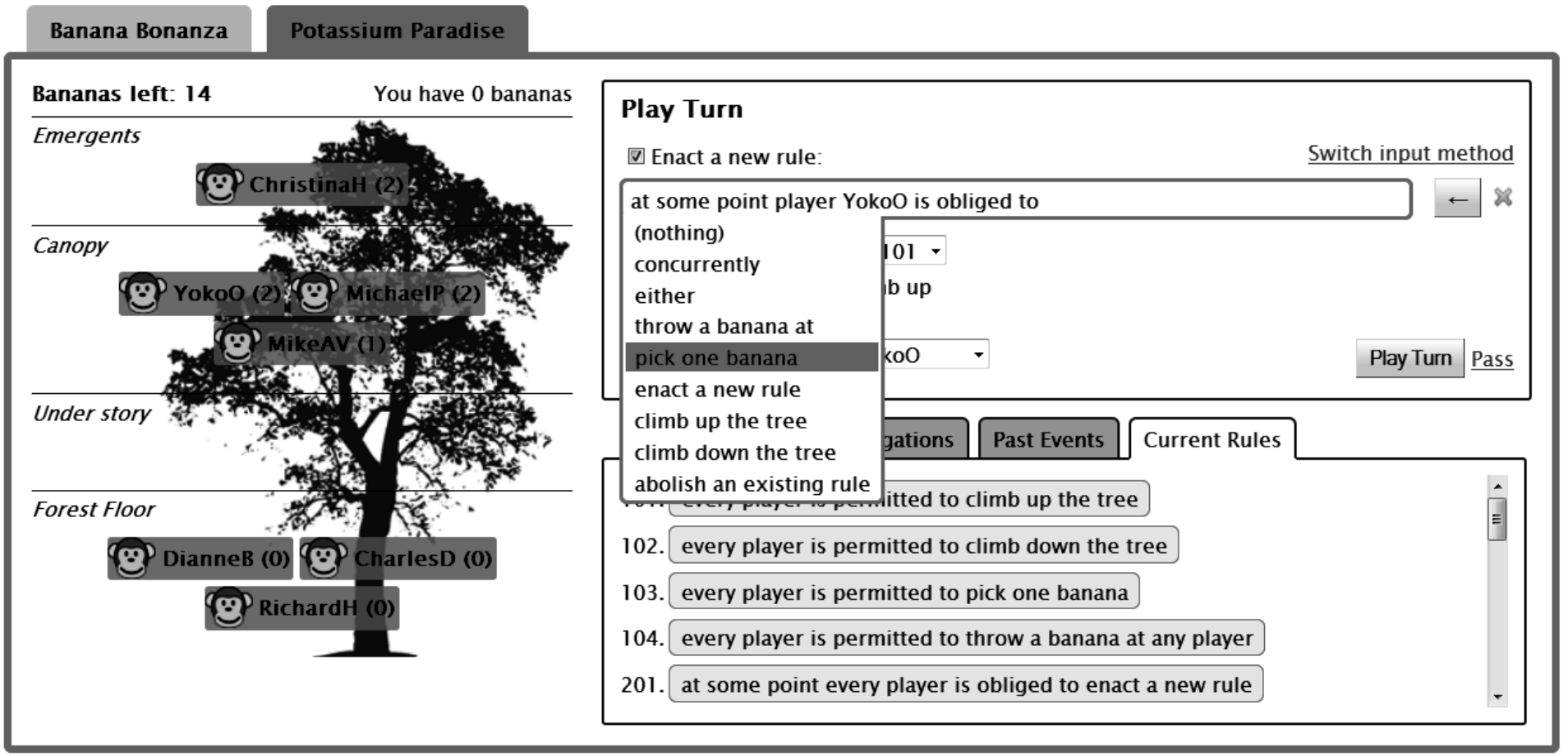
- "Fridge magnets"

  - Both closely based on examples in the Grammatical Framework distribution. http://www.grammaticalframework.org/

User interface



User interface



Evaluation

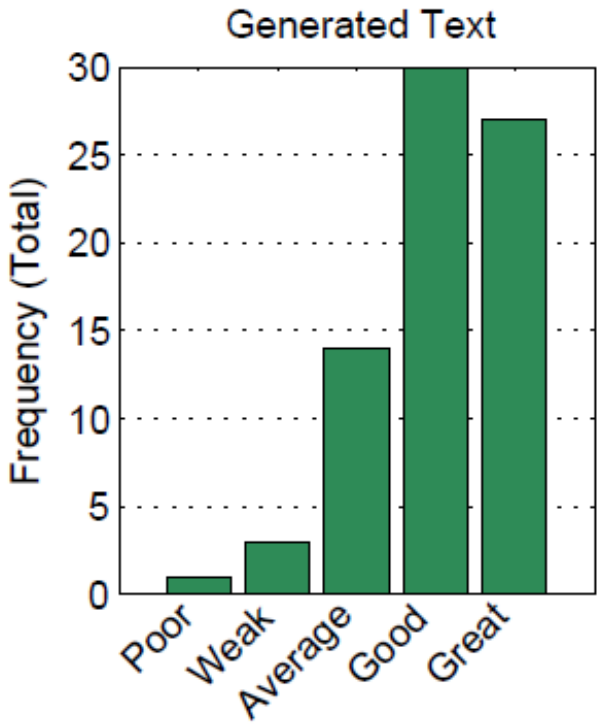
- Criteria
  1. Guided input methods
  2. Generated phrases
  3. Contract logic

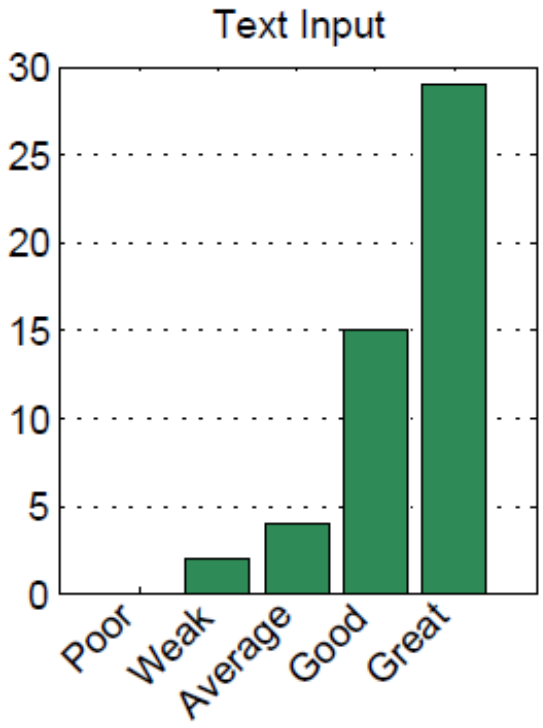
- 14 test users
- 2 concurrent games
- 9 consecutive days

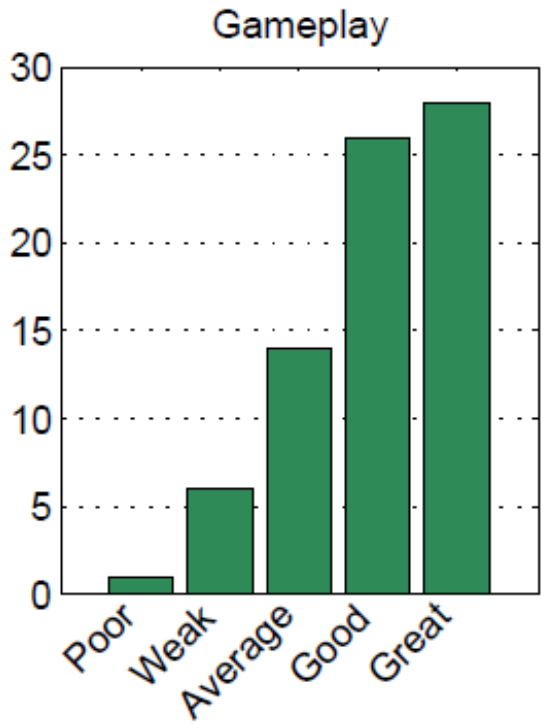
Evaluation

- In-game feedback
  - 74 responses total (1 per turn / pass)
  - 3 questions (previous criteria)
- Post-game questionnaire
  - 14 responses total (1 each)
  - 9 more general questions
- Comments

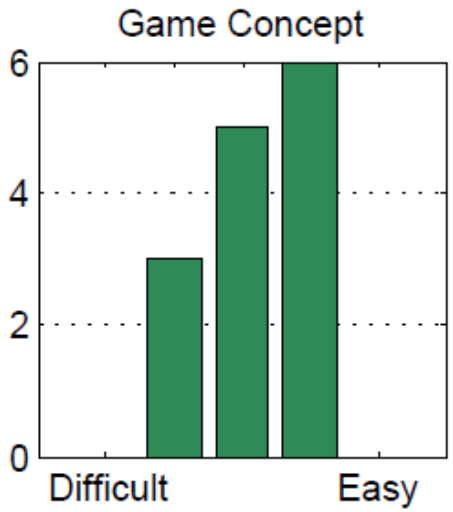
Results: In-game feedback

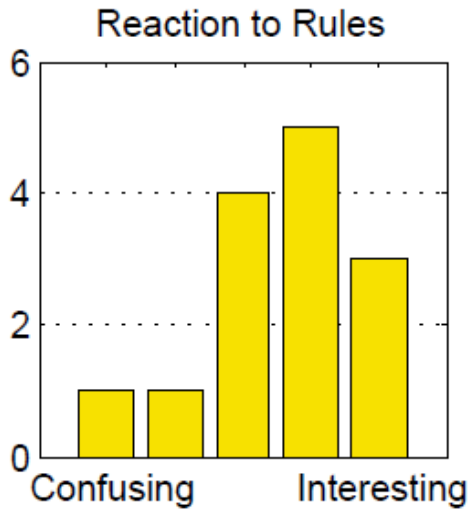


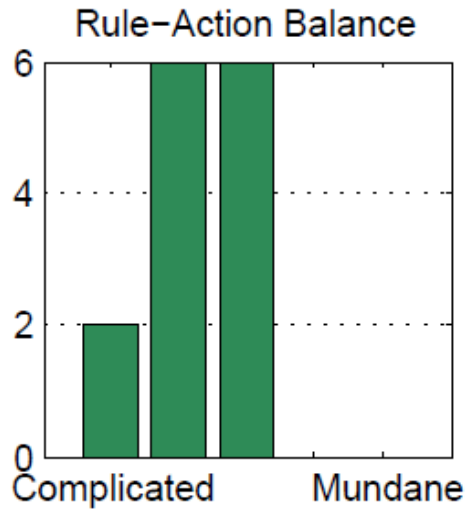


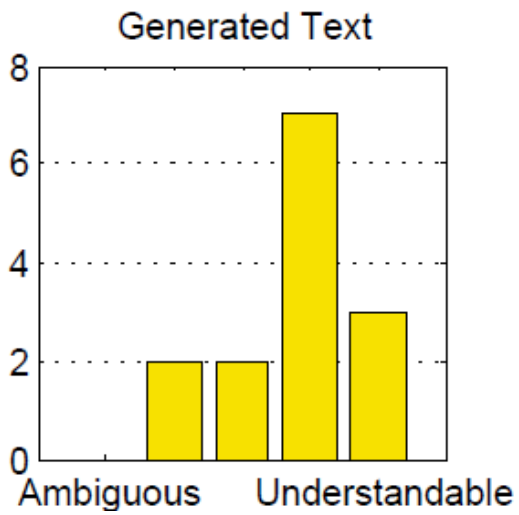


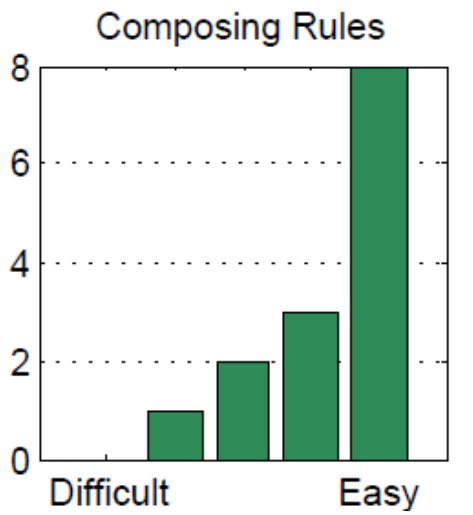
Results: Post-game questionnaire

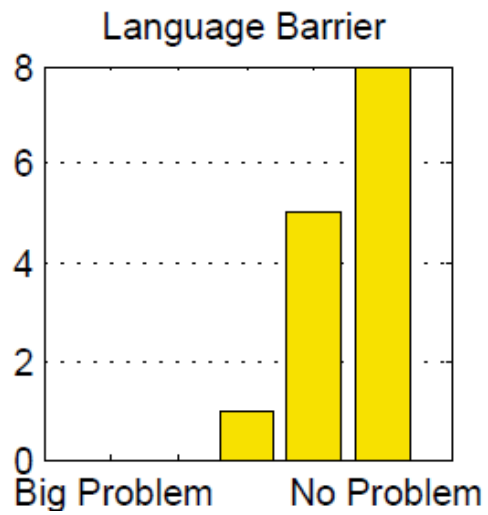












Observations

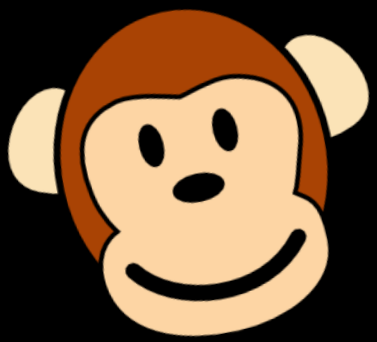
- Generated text
  - Not problematic
  - Mostly template-based
- Input methods
  - Very successful
  - 21% would prefer free-text
- Contract logic & game structure
  - Noticeably restrictive – limited adaptability
  - Ease of rule manipulation

Conclusions

- Successful within limited scope
- Limitations noted, but not prohibitive

- Future work
  - Higher adaptability (more Nomic-like)
  - Conflict / contradiction detection

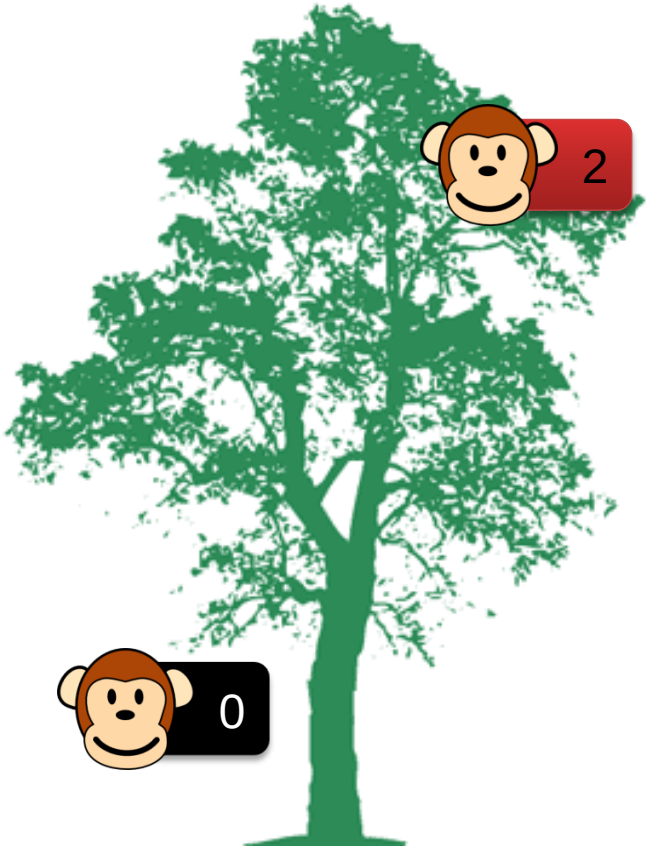
- Possible applications
  - Personal contract manager
  - Contract authoring tools



Demo:
Gameplay example

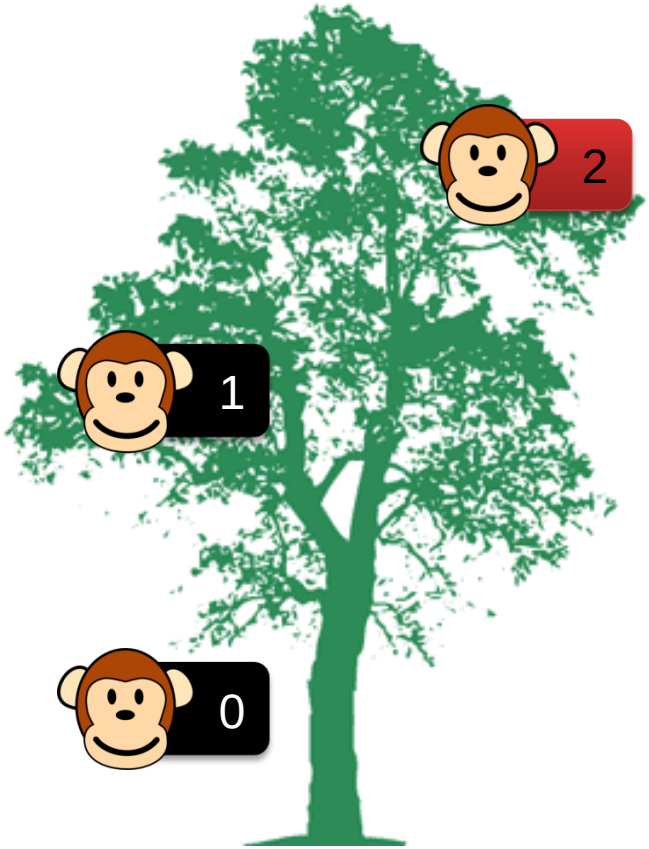
BanaNomic Demo

- RED is better off than BLACK
- Actions
  - Climb up / down
  - Pick banana
  - Throw banana
  - Enact rule
  - Abolish rule



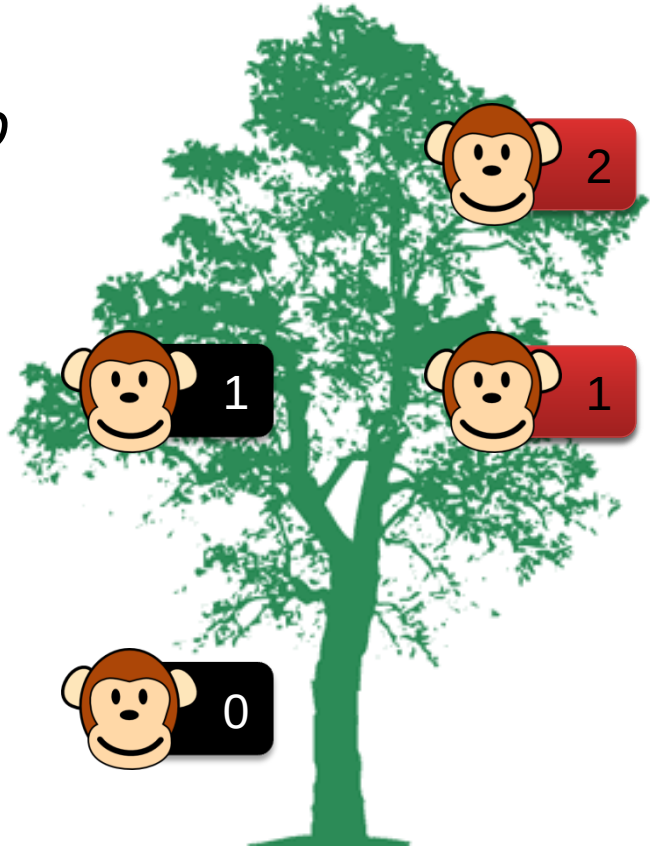
BanaNomic Demo

- BLACK
  - Climb up
  - Pick banana
- RED
  - Enact: *BLACK is prohibited from climbing up*



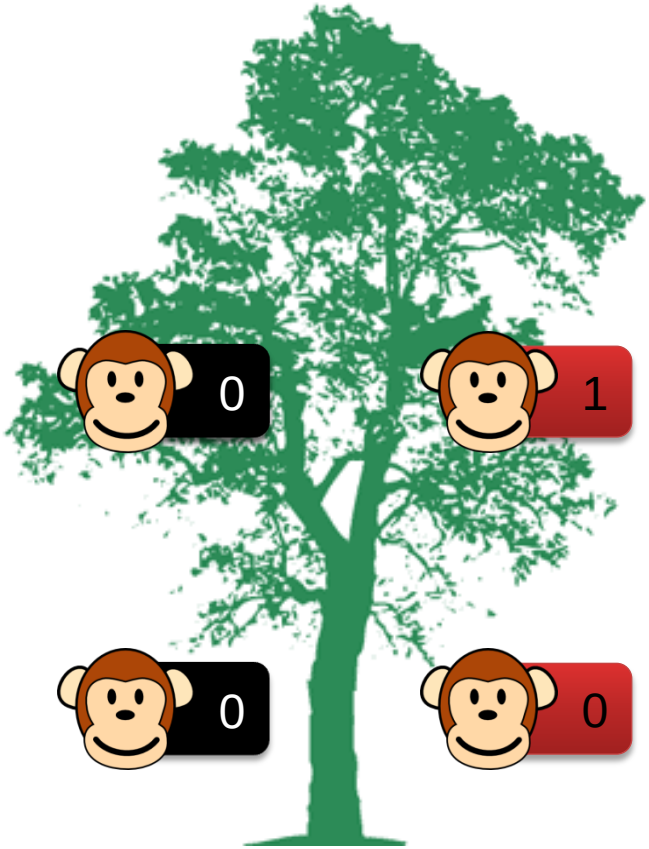
BanaNomic Demo

- BLACK
  - Enact: *RED is obliged to climb down*
- RED
  - Climb down
  - Throw banana



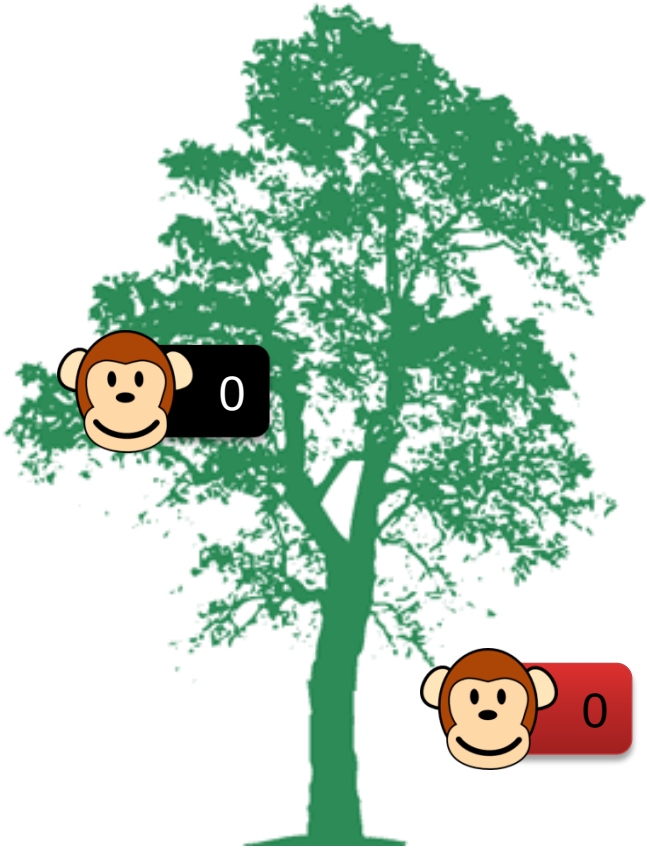
BanaNomic Demo

- BLACK
  - Climb up
  - Pick banana
  - Throw banana
  - Enact:
    - *RED is prohibited from climbing up*
    - *RED is prohibited from enacting rules*
- RED
  - Pass!



BanaNomic Demo

- Who is the winner?
  - Most points
  - Highest level
  - Most rules enacted
  - Most banana throws
  - etc.




Thank
you!