Playing Nomic using a Controlled Natural Language

John J. Camilleri Gordon J. Pace Michael Rosner

University of Malta

Objectives of the Work

Last year we spoke about a CNL for contract specification, enabling analysis of contracts . . .

- Controlled syntax aids much towards easier parsing and generation of text.
- Analysis is also possible, but it is much dependent on semantics;
- and requires knowledge of the underlying domain.
- The objective this year was therefore to study the use of a controlled natural language in a controlled semantic domain.

More Concretely. . .

- We investigate the use of a CNL to arbitrate playing Nomic, a game based on contract enforcement and amendement:
    - Design a CNL for the contracts and develop it in GF;
    - Give the language a semantics to enable exact analysis;
    - Develop a game engine arbitrating amongst players participating in the game.
- But most importantly, we limit the domain of application to enable full arbitration of the game.

Developing a CNL

- Various issues may need to be addressed when identifying a CNL:
    - Syntax of the CNL
    - Basic concepts to be combined using the language syntax (a domain context)
    - Semantics of the operators of the language
    - Semantics of the underlying domain context
    - Other information e.g. what constitutes simplicity of CNL sentences.
- The relative importance of these issues depends on the intended use:
    - Parsing: syntax, basic concepts.
    - Generation: syntax, basic concepts, other information.
    - Superficial analysis: semantics of operators.
    - Deep analysis: semantics of operators and underlying domain.
    - Feedback from analysis: requires a combination of generation and analysis.

Analysis: An Example

- Contracts written in a language with three contractual operators: obligation, permission and prohibition.
- Analysis is required to identify conflicting clauses.
  - Operator Level Conflict:

    "John is obliged to read a book."
    is in conflict with
    "John is forbidden from reading a book."

  - Domain Level Conflicts:

    "John is obliged to read a book."
    is in conflict with
    "John is forbidden from opening a book."

Contracts

- A contract can be seen as:
  1. an agreement between a number of parties;
  2. documents the regulation of their actions or behaviour.
  3. may change but according to a set of rules (possibly internally encoded).
- Contracts are sometimes seen as a set of properties that a system must satisfy, but this view does not facilitate reasoning *about* the contract e.g.
  1. How to analyse a contract into its component parts: what are the obligations and rights of the parties?
  2. At a given stage during contract execution, what are the currently undischarged obligations in the contract?
  3. Reasoning about *total* behaviour of a system including exceptional cases such as *Whenever clause (a) is violated, the user is prohibited from obtaining the service*.
- To facilitate such reasoning a contract must be regarded as a *formal object* — an instance of *a contract language*.

Games: A Controlled Domain for Contracts?

- The rules of a game can be seen as a fixed contract, regulating player behaviour.
- Game rules can be formalised.
- Hence a formalised set of game rules could provide a useful handle on the notion of a formal contract.
- Which game?

Nomic

- Nomic is a game invented by Peter Suber in 1982 *about* contract enforcement and evolution
- The rules of Nomic allow for the players to actively change the rules themselves.
- An Initial Set of rules regulates the rule-changing process.
- One substantive rule (on how to earn points toward winning); but this rule is deliberately boring so that players will quickly amend it to please themselves.

Nomic: Initial Ruleset

101. All players must always abide by all the rules then in effect, in the form in which they are then in effect. The rules in the Initial Set are in effect whenever a game begins. The Initial Set consists of Rules 101-116 (immutable) and 201-213 (mutable).

102. Initially rules in the 100's are immutable and rules in the 200's are mutable. Rules subsequently enacted or transmuted (that is, changed from immutable to mutable or vice versa) may be immutable or mutable regardless of their numbers, and rules in the Initial Set may be transmuted regardless of their numbers.

207. Each player always has exactly one vote.

209. At no time may there be more than 25 mutable rules.

213. If the rules are changed so that further play is impossible, or if the legality of a move cannot be determined with finality, or if by the Judge's best reasoning, not overruled, a move appears equally legal and illegal, then the first player unable to complete a turn is the winner.

Nomic

- In real life, Nomic rules can refer to anything:
  - Real-world: the number of buttons on the player's shirt.
  - Game-specific concepts: the number of points a player has amassed.
  - Contract-specific concepts: whether a player is currently permitted to propose a change to a particular rule.
- The challenges in implementing Nomic to be automatically arbitrated are threefold:
  - Processing user proposed contracts;
  - Enforcing the contracts;
  - Reasoning about real-life concepts.
- The only previous reasonable implementation of Nomic was PerlNomic which does not use natural language input.

BanaNomic

- BanaNomic is an attempt to reduce Nomic to a computer-regulated subset, retaining a natural language input interface.
- The players are monkeys living in a tree fighting to pick bananas and protect their stash.
- To enable computer arbitration we:
  - Eliminate references to the real-world, limiting the concepts to (i) game-internal states e.g. how high a monkey has climbed, the number of bananas it owns; and (ii) information about the state of the game e.g. John is permitted to change rule 1.
  - Restrict the input to a controlled natural language.

A Deontic Logic for the Game Rules

- Deontic logic enables the expression of and reason about notions such as permission, obligation and prohibition.
- Typically these notions are expressed using special modal operators:
  - Fp — it is forbidden that p
  - Op — it is obligatory that p
  - Pp — it is permitted that p
- The logic we develop is action-based (ought-to-do) as opposed to state-based (ought-to-be).
- It also contains a notion of time.

The Logic

- The logic is based on a limited number of well-defined **basic** actions which may be tagged with their subject and object e.g. *throwBanana(John, Gordon)*.
- Action expressions may be used to define **compound** behaviour e.g. *throwBanana(John, Gordon)* $\wedge$ *pickBanana(Mike)*.
- Deontic operators may be applied to these expressions e.g. *F(pickBanana(Paul))*.
- Deontic statements can be combined together with temporal operators e.g. $\Diamond[0, 10]$ *O(throwBanana(Gordon, John))*.
- Quantification (at the contract clause level) is allowed through the use of a placeholder $*$ e.g. *F(throwBanana($*$,John))*.

Syntax of the Logic

$$Clause \quad ::= \quad Ok \mid Fail$$

Basic Contracts

The trivially accepted contract *Ok* and trivially refuted contract *Fail*.

Syntax of the Logic

$$\begin{array}{lcl} \mathit{Clause} & ::= & \mathit{Ok} \mid \mathit{Fail} \\ & \mid & O(\mathit{ActionExpression}) \mid P(\mathit{ActionExpression}) \mid F(\mathit{ActionExpression}) \end{array}$$

Deontic Operators

Obligations, Permissions and Forbidden actions (prohibitions).

Syntax of the Logic

$$\begin{array}{rcl} \mathit{Clause} & ::= & \mathit{Ok} \mid \mathit{Fail} \\ & \mid & O(\mathit{ActionExpression}) \mid P(\mathit{ActionExpression}) \mid F(\mathit{ActionExpression}) \\ & \mid & \mathit{Clause} + \mathit{Clause} \mid \mathit{Clause} \wedge \mathit{Clause} \\ & \mid & \mathit{Clause} \lhd \mathit{Query} \rhd \mathit{Clause} \end{array}$$

Choice, Conjunction, and Conditions

Standard regular expression-like contract combinators.

Syntax of the Logic

$$\begin{array}{lcl} \mathit{Clause} & ::= & \mathit{Ok} \mid \mathit{Fail} \\ & \mid & O(\mathit{ActionExpression}) \mid P(\mathit{ActionExpression}) \mid F(\mathit{ActionExpression}) \\ & \mid & \mathit{Clause} + \mathit{Clause} \mid \mathit{Clause} \wedge \mathit{Clause} \\ & \mid & \mathit{Clause} \lhd \mathit{Query} \rhd \mathit{Clause} \\ & \mid & \mathit{Clause} \blacktriangleleft \mathit{Clause} \blacktriangleright \mathit{Clause} \end{array}$$

Sequentiality

Follow up a contract by another, depending on whether it was satisfied (sequential composition) or broken (reparation).

Syntax of the Logic

$$
\begin{array}{rcl}
\mathit{Clause} & ::= & \mathit{Ok} \mid \mathit{Fail} \\
 & \mid & O(\mathit{ActionExpression}) \mid P(\mathit{ActionExpression}) \mid F(\mathit{ActionExpression}) \\
 & \mid & \mathit{Clause} + \mathit{Clause} \mid \mathit{Clause} \wedge \mathit{Clause} \\
 & \mid & \mathit{Clause} \lhd \mathit{Query} \rhd \mathit{Clause} \\
 & \mid & \mathit{Clause} \blacktriangleleft \mathit{Clause} \blacktriangleright \mathit{Clause} \\
 & \mid & \Diamond[\mathit{Time}, \mathit{Time}]\mathit{Clause} \\
 & \mid & \Box[\mathit{Time}, \mathit{Time}]\mathit{Clause}
\end{array}
$$

Timing

Restricted temporal operators.

Syntax of the Logic

$$
\begin{array}{rcl}
\mathit{Clause} & ::= & \mathit{Ok} \mid \mathit{Fail} \\
 & \mid & O(\mathit{ActionExpression}) \mid P(\mathit{ActionExpression}) \mid F(\mathit{ActionExpression}) \\
 & \mid & \mathit{Clause} + \mathit{Clause} \mid \mathit{Clause} \wedge \mathit{Clause} \\
 & \mid & \mathit{Clause} \lhd \mathit{Query} \rhd \mathit{Clause} \\
 & \mid & \mathit{Clause} \blacktriangleleft \mathit{Clause} \blacktriangleright \mathit{Clause} \\
 & \mid & \Diamond[\mathit{Time}, \mathit{Time}]\mathit{Clause} \\
 & \mid & \Box[\mathit{Time}, \mathit{Time}]\mathit{Clause}
\end{array}
$$

Power and Delegation

- The notion of self-amendment, and power to change contract clauses is achieved by using two special actions:
    1. *enact(player, position, clause)* is the action of the player enacting a clause at the identified position in the contract.
    2. *abolish(player, position)* is the action of the player abolishing the clause at the identified position.
- Combining the deontic operators and these actions enables the expression of power information:
    - John is obliged to abolish a rule: *O(abolish(John,*))*.
    - John is not allowed to enact any rules: $\Box[0,\infty]$ *F(enact(John,* $*$, $*$*))*.
- Nesting these clauses enables us to talk about delegation e.g. John is obliged to enact a rule which forbids Mary from withdrawing rule 5.

BanaL

- We have developed BanaL — a CNL for Bananomic.
- It is designed as a NL representation for Bananomic rules and actions.
- It has been implemented in GF.
- Abstract syntax is very close to underlying logic.
- Concrete syntax permits different realisations of similar underlying logical constructs.
- For example, different natural language variants of BanaL could be created by defining a separate concrete syntax for each NL version.

Abstract Syntax - Examples

-
  ```
  C_Deontic (DE_Obliged "Paul"
    (A_Action (An_ThrowBanana "Ringo")))
  ```

-
  ```
  C_Query (Q_PointsLt "John" 5)
    (C_Deontic (DE_Permitted "John" (A_Action An_Pick)))
    C_Empty
  ```

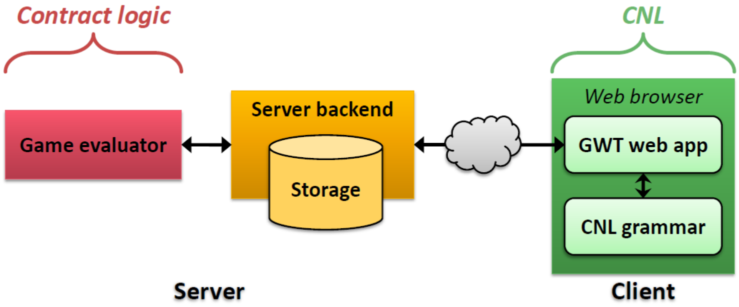
-
  ```
  C_Conditional
    (DE_Permitted "George" (A_Action An_Enact))
    C_Empty
    (C_Sometimes T_None (T_Time 9)
      (DE_Obliged "George" (A_Action An_Abolish)))
  ```

Concrete Syntax - Examples

- Paul is obliged to throw a banana at Ringo
- If John's points are less than 5 then John is permitted to pick a banana
- If George is permitted to enact a rule the (nothing) otherwise at some point before time 9 George is obliged to abolish a rule

The Implementation

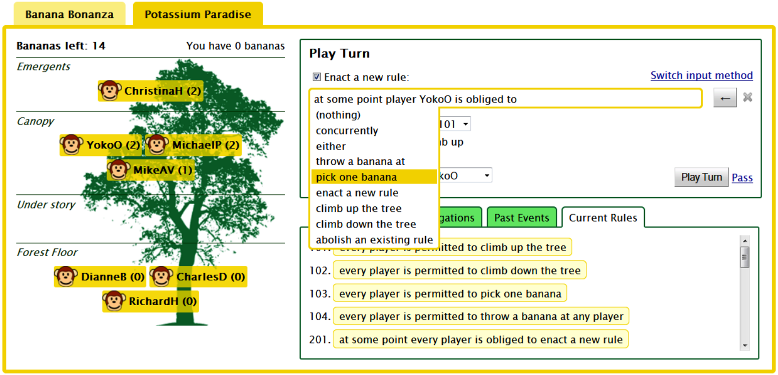
- Playable, web-based version of BanaNomic.
- Contract logic processor on the backend, built from scratch in Haskell.
- CNL interface (generation and analysis) on the frontend, using Grammatical Framework (GF).



- Players had one turn per day to:
  - Perform basic actions (climb, pick etc.)
  - Abolish / enact a rule

The Implementation

- Graphical representation of game state.
- List of current rules and effective obligations, permissions and prohibitions.
- Drop-down auto-completion for aiding the construction of grammatical CNL phrases.



Banana Bonanza | Potassium Paradise

Bananas left: 14 You have 0 bananas

Emergents
ChristinaH (2)

Canopy
YokoO (2) MichaelP (2)
MikeAV (1)

Under story

Forest Floor
DianneB (0) CharlesD (0)
RichardH (0)

Play Turn

☑ Enact a new rule: Switch input method

at some point player YokoO is obliged to

(nothing)
concurrently
either
throw a banana at
pick one banana
enact a new rule
climb up the tree
climb down the tree
abolish an existing rule

Play Turn Pass

gations | Past Events | Current Rules

...itted to climb up the tree
102. every player is permitted to climb down the tree
103. every player is permitted to pick one banana
104. every player is permitted to throw a banana at any player
201. at some point every player is obliged to enact a new rule

The CNL: Discussion

- BanaL is very close to the underlying logic.
- Main advantage is that not only is it easy to carry out the translation between CNL and logic but that this works in both directions.
- Main disadvantage is that the “success" of BanaL results from a highly delicate balance between the simplicity of the logic and the “naturalness", and hence acceptability, of the CNL.
- If the underlying logic becomes more complex, the CNL must expand in order to retain naturalness.
  - Example: bananas can be large, small, or rotten, (and have different values to the holder).
- If the CNL becomes more complex, then the underlying logic must accommodate the the enhanced semantic potential.
  - The CNL includes numbers (e.g. “three bananas").
- There is no end to the ways in which the CNL might be expanded.

What Counts as Natural in CNLs?

- We are accustomed to assuming that CNLs are simply reduced forms of NLs – simpler but essentially made of the same stuff.
- It may be interesting to speculate on the use of “stuff" which is natural, which has a direct bearing on semantics, but is not normally considered as part of natural language.
- The elements underlying web-page styles and web-page design.
    - Examples include: headings, lists, indentation, highlighting etc.
- Formalisation is not the issue.
- The issue is the largely unexplored relation between the layout and the semantics, which is made use of, in an informal way, by e.g. legislators (legal documents) and mathematicians (proofs).
- Is this an area in which CNLs might profitably be applied?

The Semantics: Discussion

- Despite the controlled semantic domain, there are still many concepts not encoded in the contracts and enforced e.g. order of play.
- Can these be all made explicit?
- What about the semantics of the language?
  - PerlNomic enforces the semantics of Perl on the contracts.
  - We implicitly enforce an operational semantics of the contract language.
  - Can the semantics be encoded in the contract — making them also mutable according to the contract itself? Does this self-referentiality lead to any contradictions?