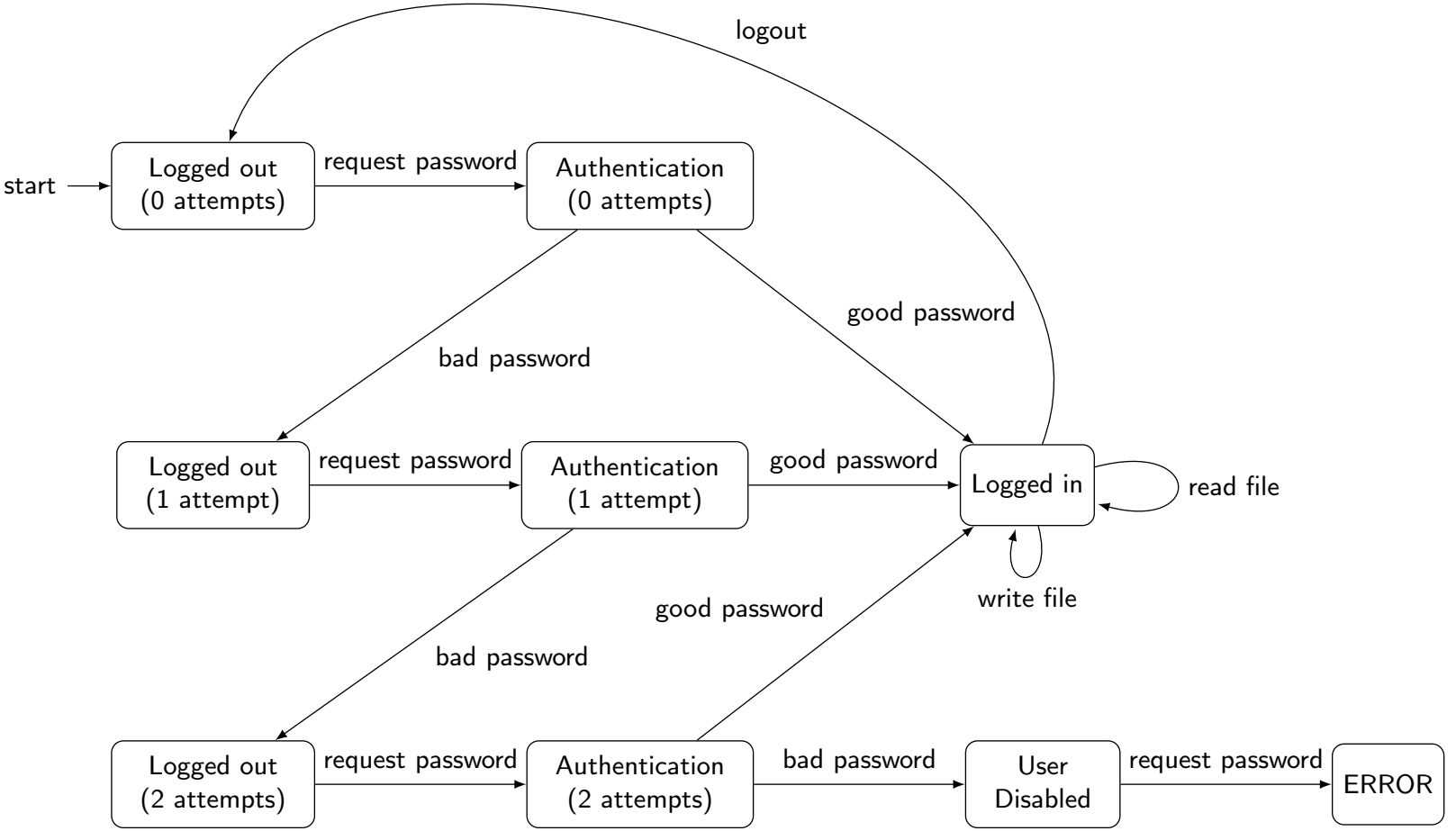
Explaining Violation Traces with Finite State Natural Language Generation Models

Gordon J. Pace Michael Rosner

University of Malta

August 2014


logout
start
Logged out
(0 attempts)
request password
Authentication
(0 attempts)
bad password
good password
Logged out
(1 attempt)
request password
Authentication
(1 attempt)
good password
Logged in
read file
write file
good password
bad password
Logged out
(2 attempts)
request password
Authentication
(2 attempts)
bad password
User
Disabled
request password
ERROR

Reporting a trace

Dear Developer,

This is what happened...

login()
badPasswd()
⋮
requestPasswd()

Can you please fix?

Regards,
G

Reporting a trace

Dear System-Architect,

This is what happened:

1. The user requested to log in, gave a correct password and after reading from a file logged out.
2. The user requested to log in, and gave a bad password.
3. After a log in request the user gave a correct password and wrote twice to a file before logging out.
4. ...
5. Finally, the user made a request to log in, which should not have been allowed.

Can you check whether this is a bug, or the rule requires updating?

Regards,
G

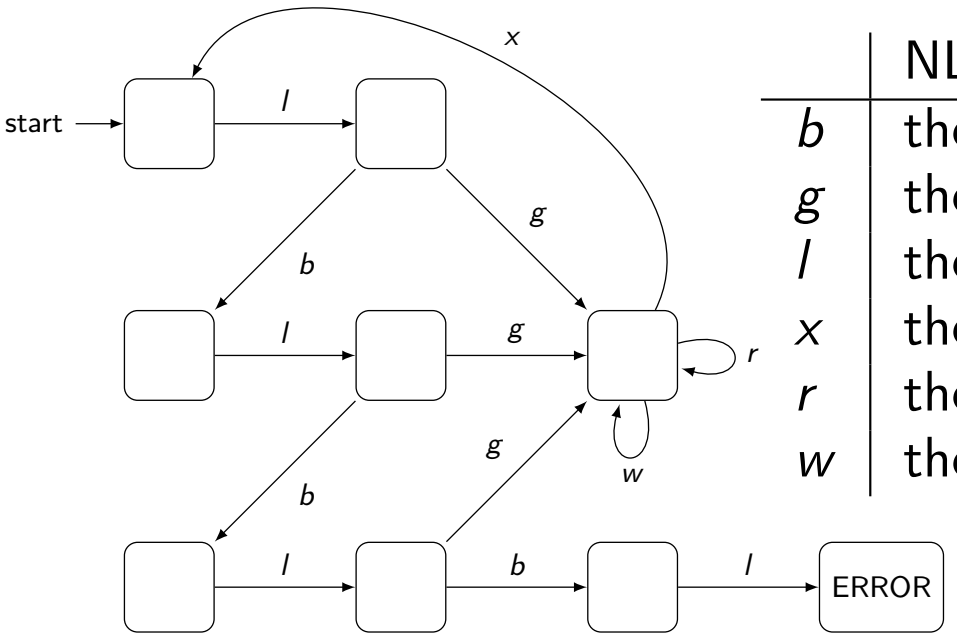
Reporting a trace

Dear Rule-Manager,

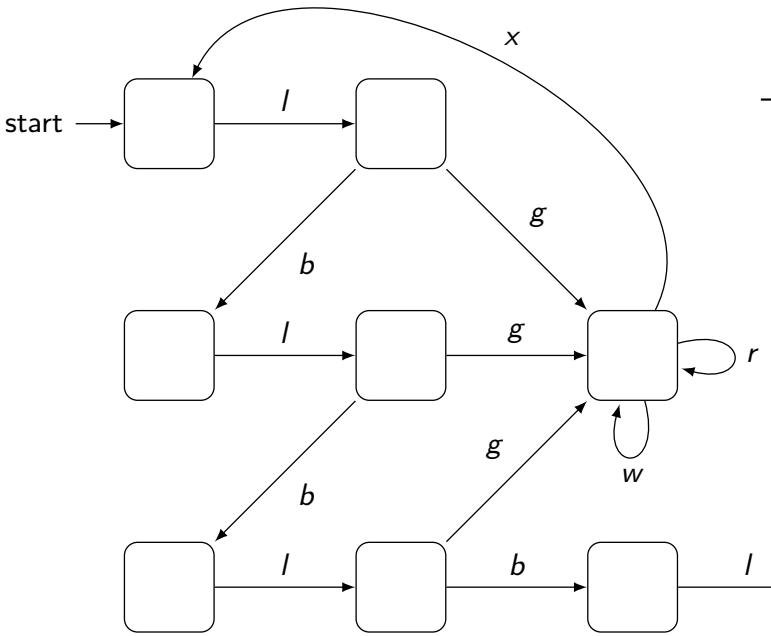
This is what happened: *The user logged in a number of times, interspersed by sequences of one or two bad logins, after which she unsuccessfully attempted to log in 3 times. The user then made another request to log in, which should not have been allowed.*

Can you please confirm that the rule has fired as expected?

Regards,
G



	NL explanation
b	the user gave a bad password
g	the user gave a good password
l	the user requested to log in
x	the user logged out
r	the user read from a file
w	the user wrote to a file



	NL explanation
b	the user gave a bad password
g	the user gave a good password
l	the user requested to log in
x	the user logged out
r	the user read from a file
w	the user wrote to a file

lgrxlblgwwxlgrwxlgxlblblbl

A naïve explanation of the trace: CNL0

The user requested to log in. The user gave a good password. The user read from a file. The user logged out. The user requested to log in. The user gave a bad password. The user requested to log in. The user gave a correct password. The user wrote to a file. The user wrote to a file. The user logged out. The user requested to log in. The user gave a correct password. The user read from a file. The user wrote to a file. The user logged out. The user requested to log in. The user gave a good password. The user logged out. The user requested to log in. The user gave a bad password. The user requested to log in. The user gave a bad password. The user requested to log in. The user gave a bad password. The user requested to log in, which should not have been allowed.

Structured Dictionary

b	*user*	*gives*	*bad password*
l	*user*	*requests*	*login*
g	*user*	*gives*	*good password*
x	*user*	*logs*	*out*
r	*user*	*reads*	*from a file*
w	*user*	*writes*	*to a file*

A grouped explanation: CNL1

1. The user requested to log in. The user gave a good password. The user read from a file. The user logged out.
2. The user requested to log. The user gave a bad password.
3. The user attempted to log in. The user gave a good password. The user wrote to a file. The user wrote to a file. The user logged out.
4. The user requested to log in. The user gave a good password. The user read from a file. The user wrote to a file. The user logged out.
5. The user requested to log in. The user gave a good password. The user logged out.
6. The user requested to log in. The user gave a bad password.
7. The user requested to log in. The user gave a bad password.
8. The user requested to log in. The user gave a bad password.
9. The user requested to log in, which should not have been allowed.

Adding Structural Information: CNL1

- Grouping described using regular expressions:
  Correct login session: $lg(r + w)^*x$.
  Sequence of incorrect login requests: $(lb)^*$.
- Enumeration achieved through the use of output to a language such as HTML or LaTeX.

A better grouped explanation: CNL2

1. The user requested to log in, gave a correct password and after reading from a file logged out.
2. The user requested to log in, and gave a bad password.
3. After a log in request the user gave a correct password and wrote twice to a file before logging out.
4. The user requested to log in, gave a correct password, read from a file, wrote to a file and then logged out.
5. After requesting a log in, the user gave a good password and logged out.
6. The user requested to log in, gave a bad password, requested again to log in, gave another bad password and after requesting to log in, gave another bad password.
7. Finally, the user made a request to log in, which should not have been allowed.

Implementing CNL2

- CNL2 features:
  1. Punctuation other than full stops
  2. Temporal connectives (“after”, “then”, “finally”)
  3. The use of contrastive conjunctions like “but”
  4. Collective terms (“twice”)
- Special rules are used to aggregate sequences of events using proper connectives (easy to generalise).
- Domain and property information is used to add contrastive conjunctions *but* and connectives such as *finally* (less easy to generalise).

A more natural explanation: CNL3

The user logged in a number of times, interspersed by sequences of one or two bad logins, after which she unsuccessfully attempted to log in 3 times. The user then made another request to log in, which should not have been allowed.

Adding Abstraction: CNL3

Consecutive correct login sessions: $(lg(r + w)^*x)^n$ explained as *"The user successfully logged in n times"*.

Consecutive correct failed login attempts: $(lb)^n$ explained as *"The user unsuccessfully attempted to log in n times"*.

Correct login sessions interspersed with occasional incorrect one: $((lg(r + w)^*x) * lb(lg(r + w)^*x)+)^*$ explained as *"The user successfully logged in a number of times, with one off bad logins in between"*.

Correct login sessions interspersed with occasional incorrect one or two: $((lg(r + w)^*x) * (lb + lblb)(lg(r + w)^*x))^*$ explained as *"The user logged in a number of times, interspersed by sequences of one or two bad logins"*.

Adding Contextuality

Action	Pre	Post	CNL rendering
x	$l\,\overline{x}^*$	–	user logs out
	otherwise		user attempts to log out
l	–	b	the user attempts to log in
	otherwise		the user logs in

Extending Contextuality

This technique can be further extended and refined to deal with repetition of actions as shown below with repeated logins:

Action	Pre	Post	CNL rendering
l	$l\,b\,\bar{l}^*$	b	user attempts to log in again
	–	b	user attempts to log in
	$l\,\bar{b}\,\bar{l}^*$	–	user logs in again
	otherwise		user logs in

Directions: Automating the Transformation

- User input is necessary for the regular expressions for abstraction and context, and the dictionary.
- Can the rest can all be automated?
- Can dictionaries be used across properties?

Directions: Richer Properties and Traces

- Properties may have richer information which cause violation (unlike our simple example which just has an *error* state e.g. an access policy may be violated by not making available a file which the user should have access to, or by making available a file which the user should be forbidden from accessing.
- In some cases, availability of system state per event in the trace may be required for full explanations e.g. what was the user account balance when she attempted to transfer the money?

Directions: Properties as Classifiers

- Properties may be used to classify users based on logs (e.g. fraudulent users).
- Explanation may include why a user falls under that category.

Directions: Agents and Causality

- Multiple agents (e.g. users in our example) may be supporting by "unzipping" the trace into subtraces performed by (or on) different agents.
- Causality information may be required (e.g. a file may not be found because another user deleted it).

Directions: Automated Extraction of Abstractions

- Frequent substring analysis (string matching problem).
- Looking into frequent regular expression which match on subsequences (grammatical inference problem).

Directions: Relative and Real Time

- Timestamps to allow for identification of points in time and the notion of time passing.
- Intervals as context.

Summary and Conclusions

- We have been looking into means of supporting text generation over a CNL specific to particular events and properties.
- Are finite state techniques sufficient to obtain reasonably natural text starting from a näive generation approach — essentially bridging the gap between the controlled and the natural?
- Why finite state techniques?
  - Theory well understood
  - Efficient
  - Familiar notation to specify abstractions/context
- The real question is: How far can we push this approach before a richer specification language gives better returns?