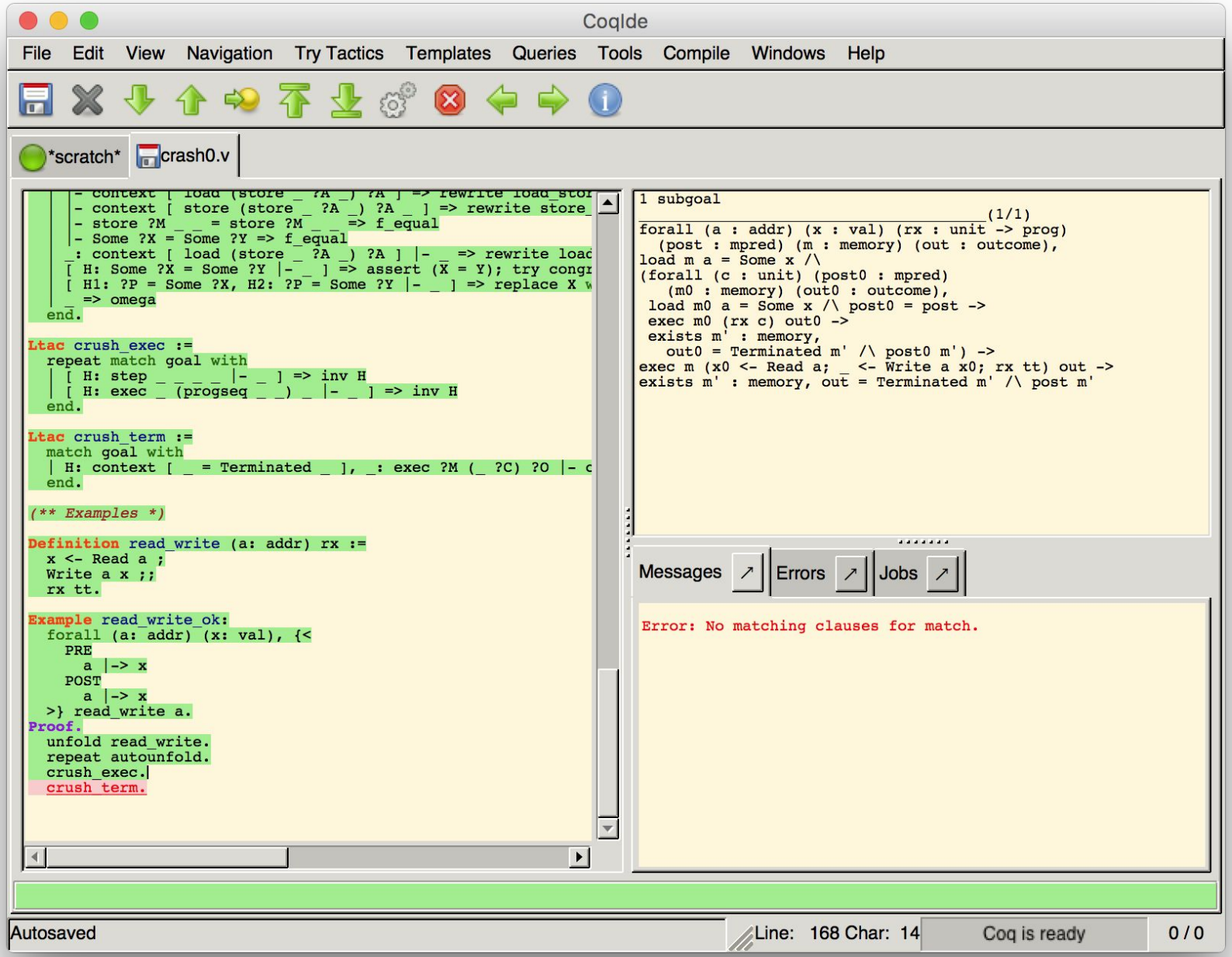
ProverBot9000

A proof assistant assistant

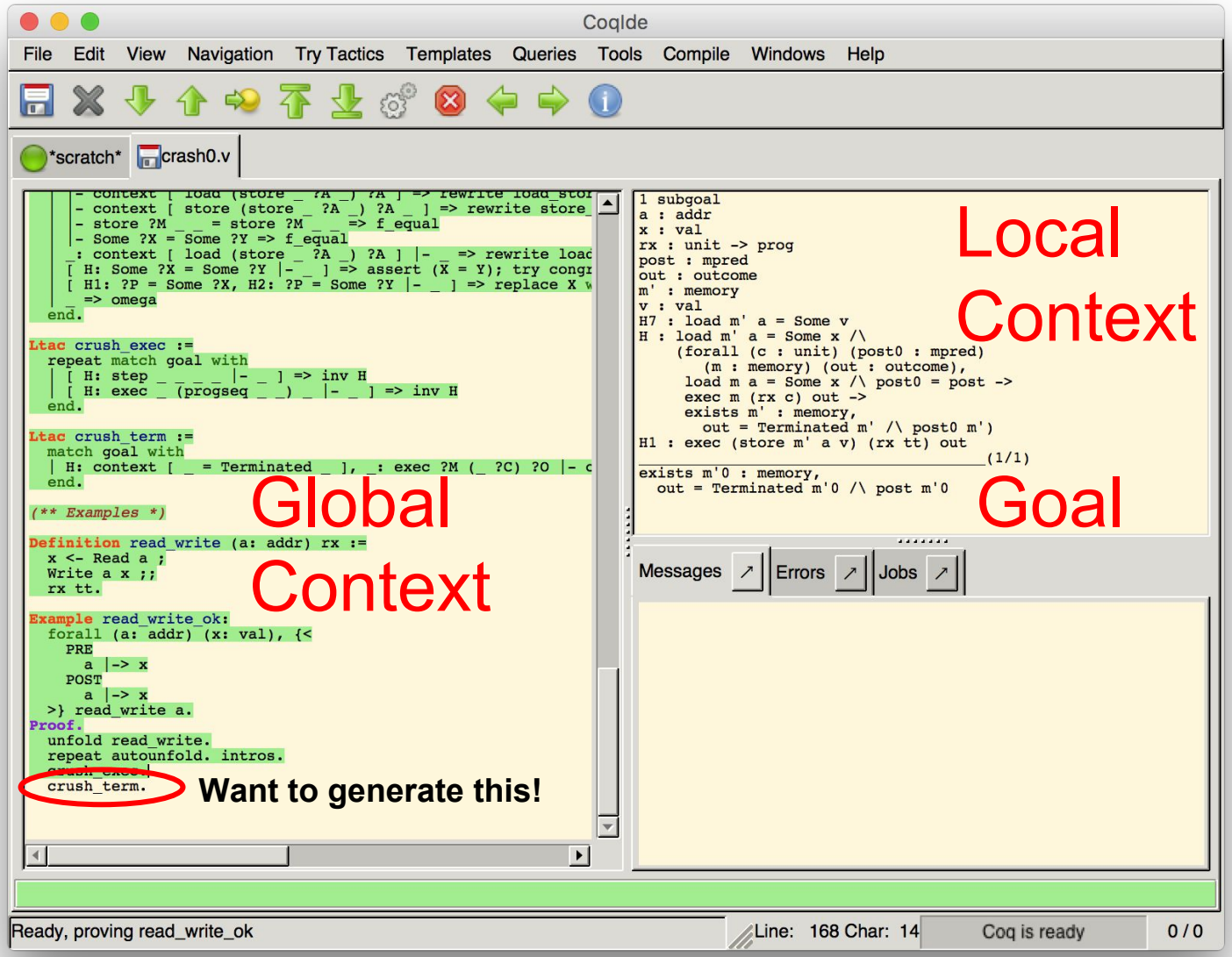
Proofs are hard


CoqIde
File Edit View Navigation Try Tactics Templates Queries Tools Compile Windows Help
scratch
crash0.v
| - context [load (store _ ?A _) ?A] => rewrite load_stor
| - context [store (store _ ?A _) ?A _] => rewrite store_
| - store ?M _ _ = store ?M _ _ => f_equal
| - Some ?X = Some ?Y => f_equal
_: context [load (store _ ?A _) ?A] |- _ => rewrite loac
[H: Some ?X = Some ?Y |- _] => assert (X = Y); try congr
[H1: ?P = Some ?X, H2: ?P = Some ?Y |- _] => replace X w
_ => omega
end.
Ltac crush_exec :=
repeat match goal with
| [H: step _ _ _ _ |- _] => inv H
| [H: exec _ (progseq _ _) _ |- _] => inv H
end.
Ltac crush_term :=
match goal with
| H: context [_ = Terminated _], _: exec ?M (_ ?C) ?O |- c
end.
(** Examples *)
Definition read_write (a: addr) rx :=
x <- Read a ;
Write a x ;;
rx tt.
Example read_write_ok:
forall (a: addr) (x: val), {<
PRE
a |-> x
POST
a |-> x
>} read_write a.
Proof.
unfold read_write.
repeat autounfold.
crush_exec.
crush_term.
1 subgoal
(1/1)
forall (a : addr) (x : val) (rx : unit -> prog)
(post : mpred) (m : memory) (out : outcome),
load m a = Some x /\
(forall (c : unit) (post0 : mpred)
(m0 : memory) (out0 : outcome),
load m0 a = Some x /\ post0 = post ->
exec m0 (rx c) out0 ->
exists m' : memory,
out0 = Terminated m' /\ post0 m') ->
exec m (x0 <- Read a; _ <- Write a x0; rx tt) out ->
exists m' : memory, out = Terminated m' /\ post m'
Messages
Errors
Jobs
Error: No matching clauses for match.
Autosaved
Line: 168 Char: 14
Coq is ready
0 / 0

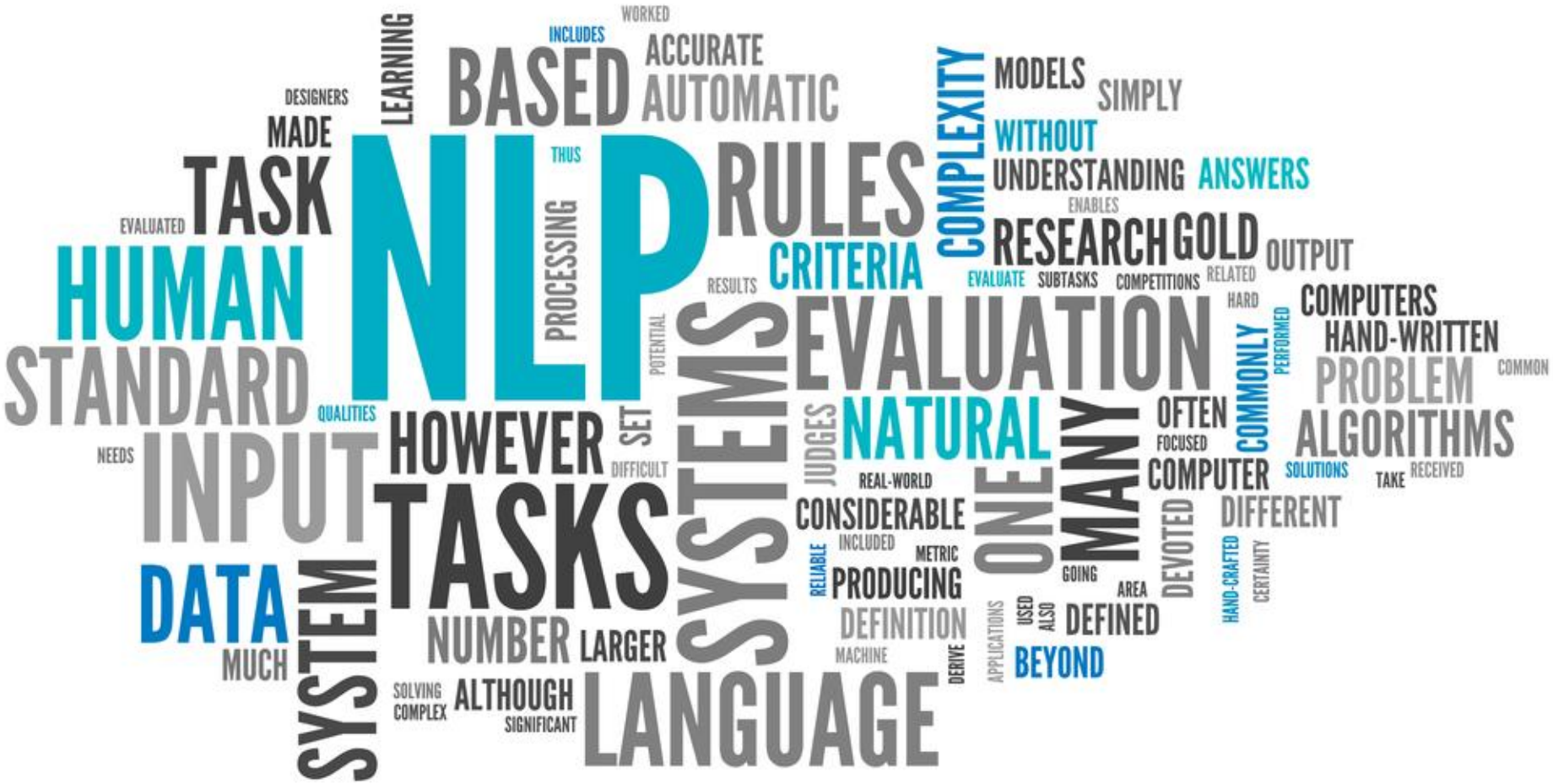
Proof assistants are hard

Big Idea: Proofs are hard, make computers do them

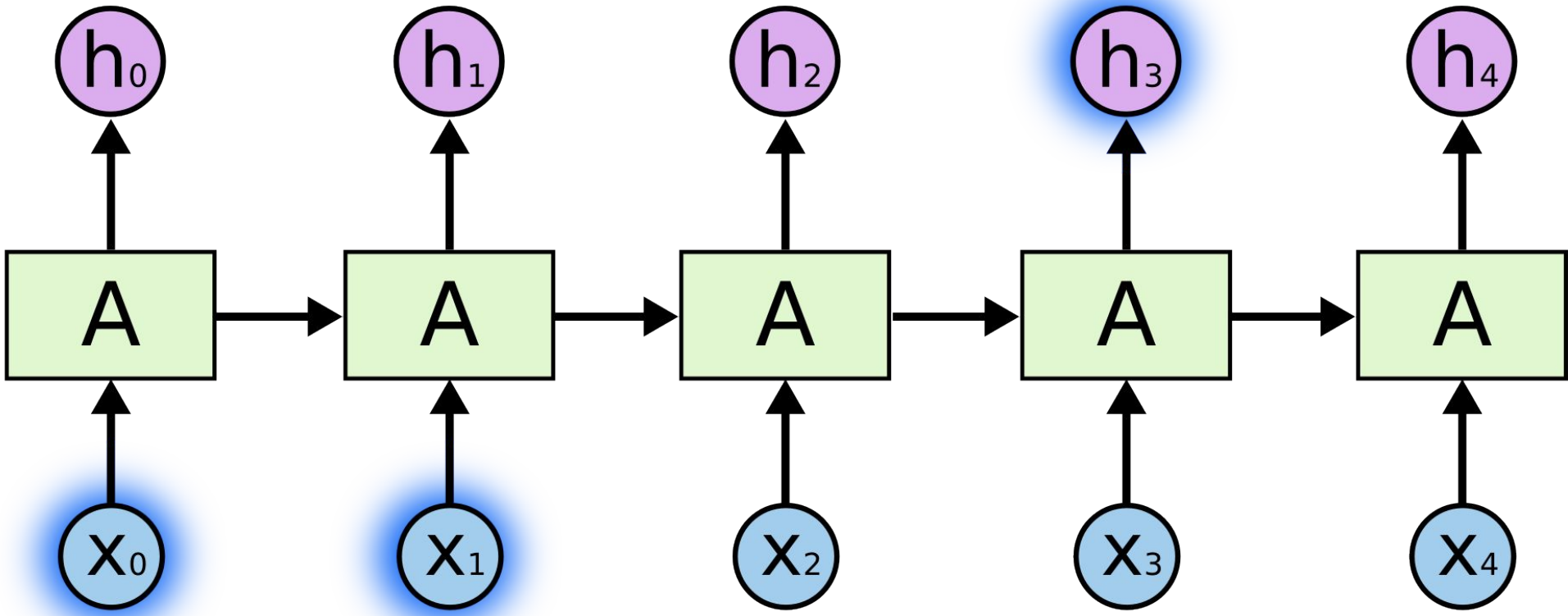
Proofs are just language with lots of structure



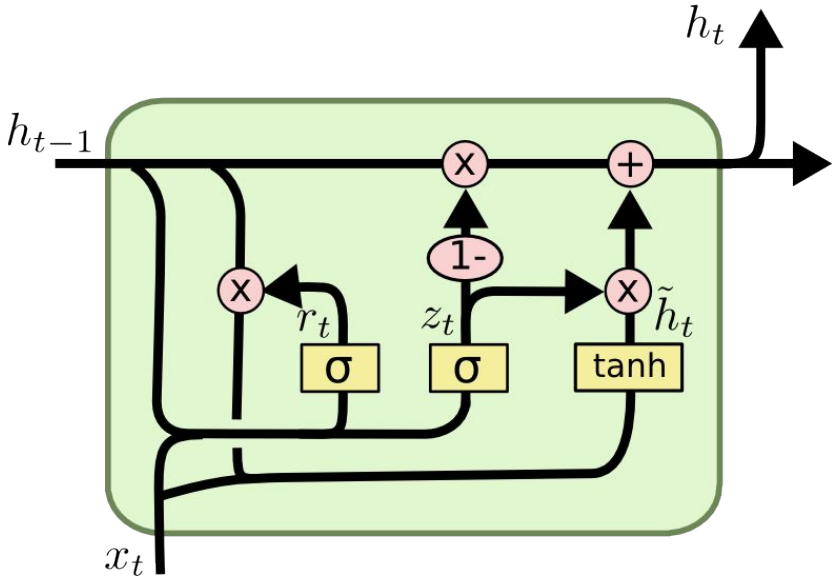
NLP techniques are good at modelling language



We use RNNs to model the “language” of proofs



We use GRUs for internal state updates



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Probably good idea: Tokenize proofs “smartly”

Works well with english:

“The quick brown robot reaches for Doug’s neck…”

->

<tk9> <tk20> <tk36> <UNK> <tk849> <tk3> ….

Custom proof names and tactics make this hard:

AppendEntriesRequestLeaderLogs
OneLeaderLogPerTerm
LeaderLogsSorted
RefinedLogMatchingLemmas
AppendEntriesRequestsCameFromLeaders
AllEntriesLog
LeaderSublog

Easy, bad idea: Model proofs char by char

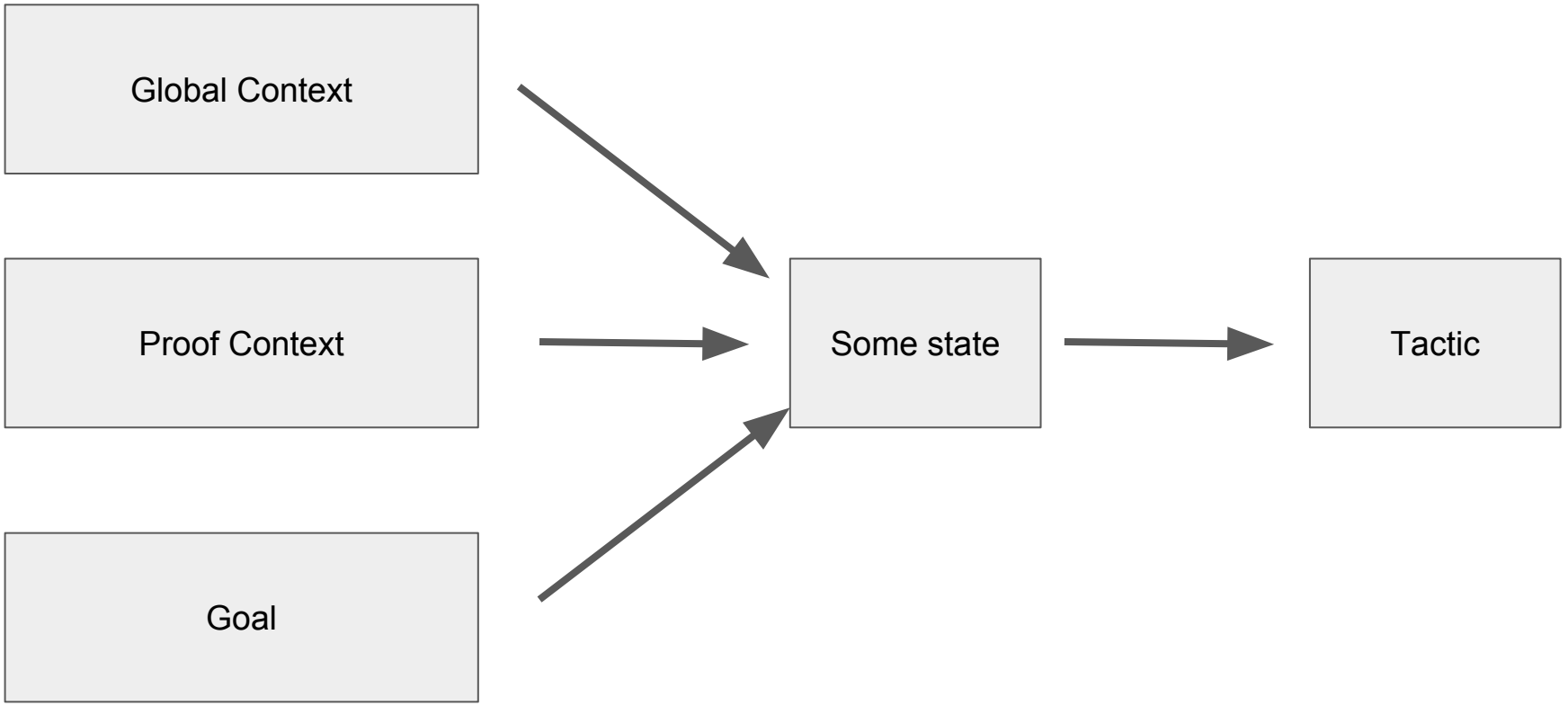
Pros:

- Very general, can model arbitrary strings
- No “smart” pre-processing needed

Cons:

- Need to learn to spell
- Need bigger models to handle generality
- Need more training data to avoid overfitting
- Longer-term dependencies are harder, terms are separated by more “stuff”

Probably good idea: multi-stream models



Problem: during training, have to bound number of unrolled time steps. The contexts can get much larger than the space that we have to unroll time steps

Our problem formulation, one unified stream

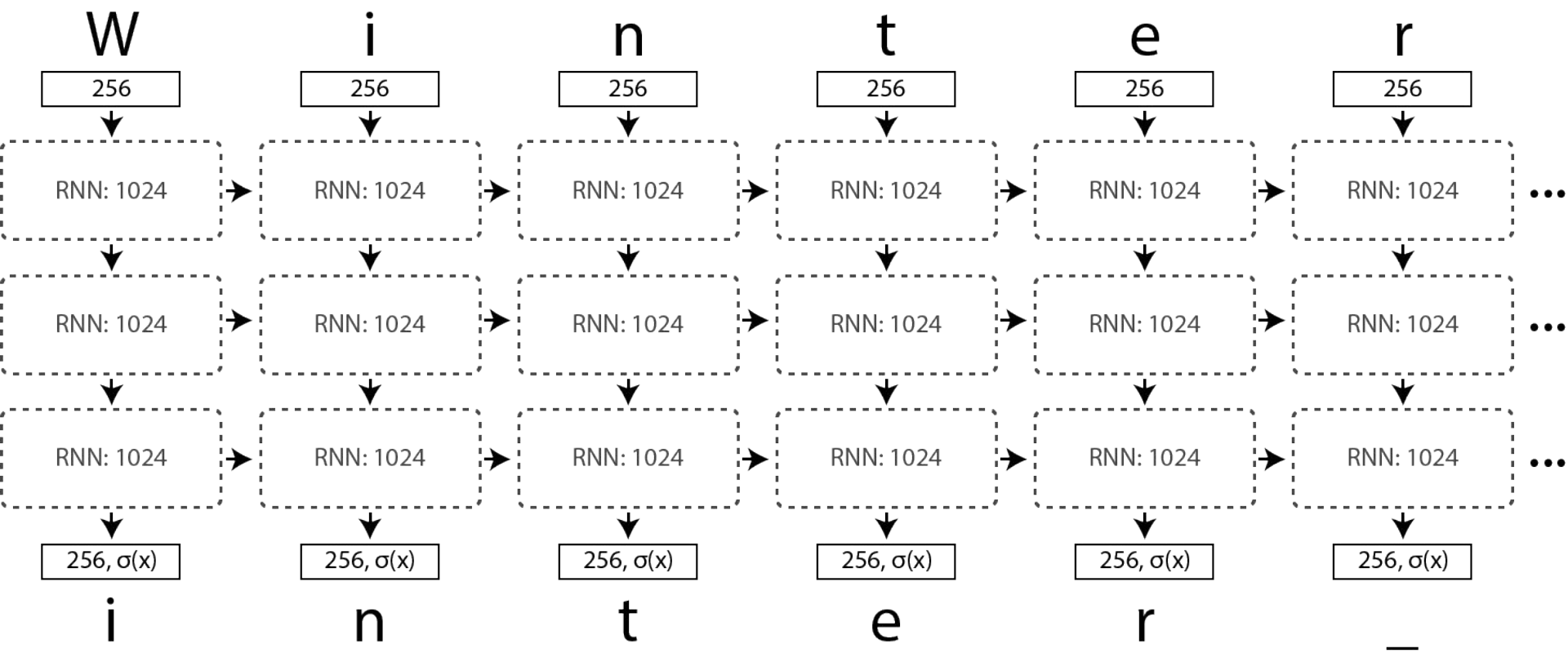
```
%%%%%
name peep_aiken_6 p.
unfold aiken_6_defs in p.
simpl in p.
specialize (p c).
do 3 set_code_cons c.
set_code_nil c.
set_instr_eq i 0%nat aiken_6_example.
set_instr_eq i0 1%nat aiken_6_example.
set_instr_eq i1 2%nat aiken_6_example.
set_int_eq n eight.
+++++
option StepEquiv.rewrite
*****
set_ireg_eq rd rd0.

…….
```

Start tokens
Previous tactics

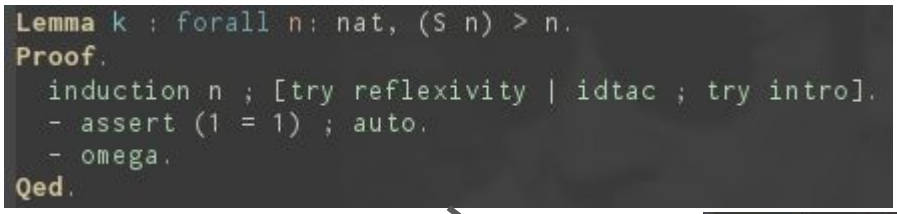
Dividing tokens
Current goal
Dividing tokens
Next tactic

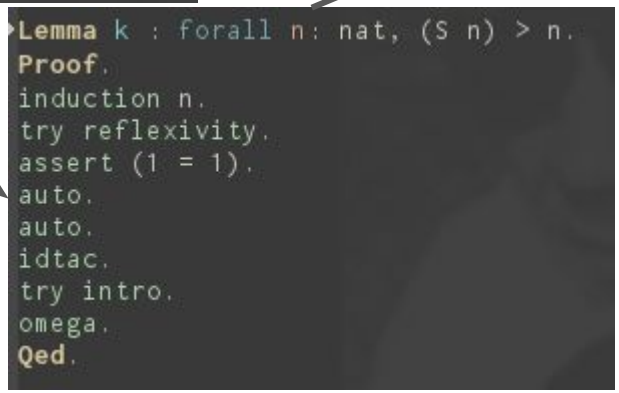
Our full model

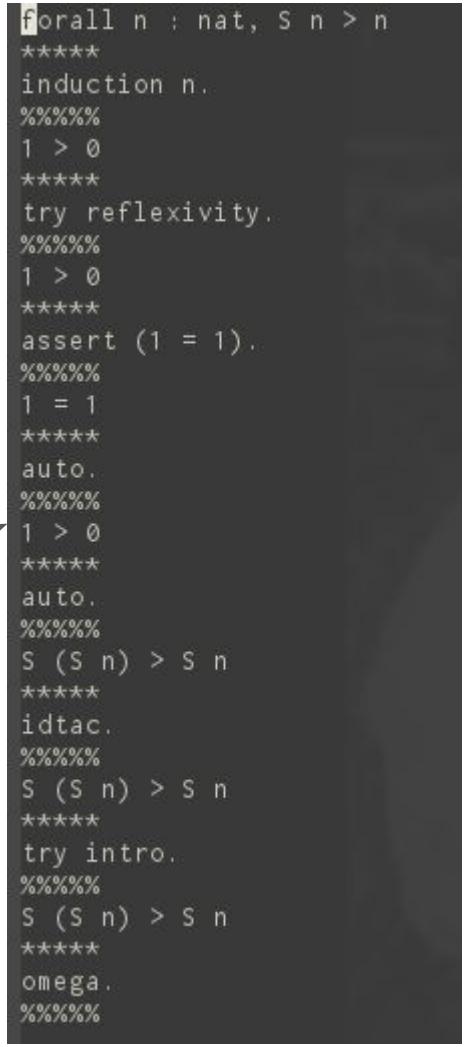


Data Extraction

- Proverbot9000 predicts tactics based on the just current goal (for now)
- Proverbot900 is trained on the Peek/Compcert codebase.
- 657 lines of python code to drive Coqtop and extract proof state
- Subgoal focusing and semicolons make proof structure more variable and complex
- We have systems which remove subgoal focusing, and from the proofs





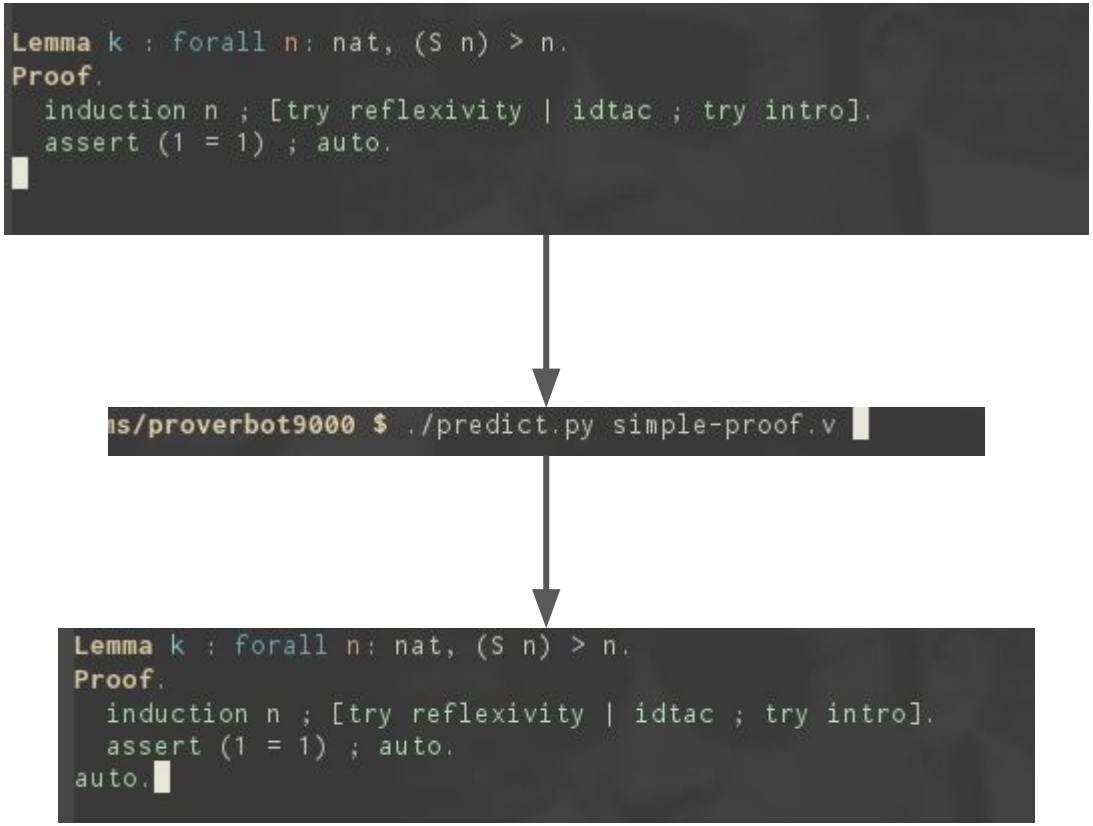


Evaluation

Our current model gets 21% accuracy on a held out set of 175 goal-tactic combinations in Peek, (aiken 5 and 6)

Interface

- Partially complete a proof
- Run proverbot
- Get a new tactic!



No subgoals left!

DEMO