An interactive driver for goal directed proof strategies

Andrea Asperti and Enrico Tassi

Department of Computer Science, University of Bologna

22 August 2008

Context

What we had:

- Matita is an ITP developed at the University of Bologna, similar to Coq
- Automation is a desirable aid, we decided to add some proof searching facility
- Non-decision procedures are usually black boxes
  - Newcomers learn nothing using automation
  - Script breakage may make automation a non-option on large developments

Aim

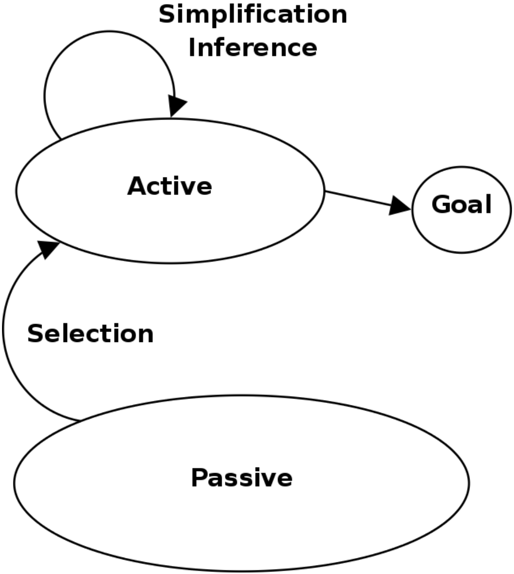
What we want:

- An automatic proof searching procedure
  - reasonably fast (not the main aim, but is fun to optimise it!)
  - designed with interactiveness in mind
- Interface to display the ongoing proof search
- Interface to drive it at runtime
- Experimentation, Debugging, Didactical purposes, ...
- A procedure producing proof scripts (not only proof objects)

Outline

- Interactive proof search procedure
- LSD resolution and ITP
- The stack issue
  - Operational description of the proof search procedure
- The interface
- Conclusion

Which kind of procedure is better suited for interactiveness? (1/4)



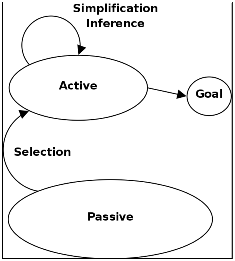
Which kind of procedure is better suited for interactiveness? (2/4)

Forward reasoning techniques are nice:

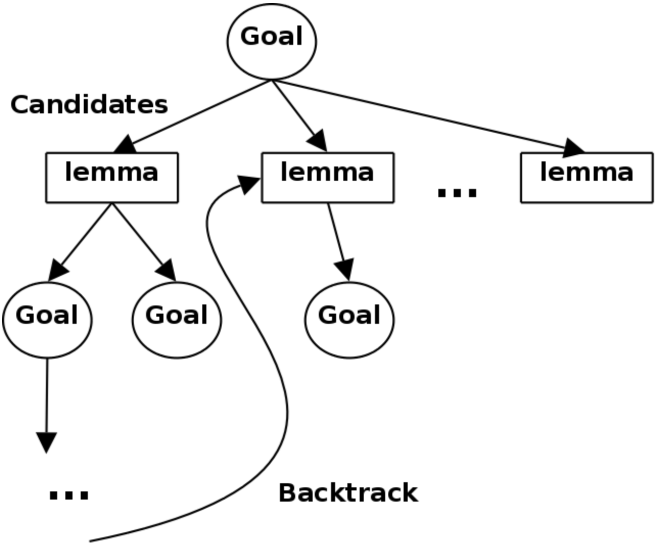
- Clear interaction point
- Fast algorithms

But we decided not to develop a procedure performing forward reasoning because:

- Not the way ITPs are used
- Active set not that stable
- Both sets are usually very big



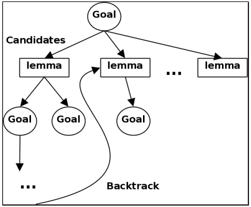
Which kind of procedure is better suited for interactiveness? (3/4)



Which kind of procedure is better suited for interactiveness? (4/4)

Depth first proof search:

- Upper part of the tree is stable
- Clear interaction point (selection)
- Failed computations can be hidden
- Goals are almost self contained



We believe that with a depth first goal directed procedure the user needs less information to follow the procedure.

SLD resolution

SLD

$$\frac{\leftarrow A_1, \ldots, A_n \qquad H \overset{c}{\leftarrow} B_1, \ldots, B_m \qquad \Sigma = mgu(H, A_i)}{\leftarrow \Sigma(A_1, \ldots, A_{i-1}, B_1, \ldots, B_m, A_{i+1}, \ldots, A_n)}$$

Apply tactic

Apply-tac

$$\dfrac{\begin{array}{l}\mathcal{P} = \Gamma_1 \vdash ?_1 : A_1, \ldots, \Gamma_n \vdash ?n : A_n \\ \mathcal{P}' = \mathcal{R}(\Gamma \vdash ?_{B_1} : B_1, \ldots \Gamma, x_1 : B_1, \ldots, x_{m-1} : B_{m-1} \vdash ?_{B_m} : B_m); \mathcal{P} \\ \Gamma \vdash c\ ?_{B_1}\ \ldots\ ?_{B_m} : H \\ \mathcal{P}',\ \Sigma,\ \Gamma \vdash H \stackrel{?}{\equiv} A_i \quad \stackrel{\mathcal{U}}{\rightsquigarrow} \quad \mathcal{P}'',\ \Sigma' \\ \Sigma'' = ?_i := c\ ?_{B_1}\ \ldots\ ?_{B_m}; \Sigma'\end{array}}{(\Sigma''(\mathcal{P}''), \Sigma'')}$$

The stack issue

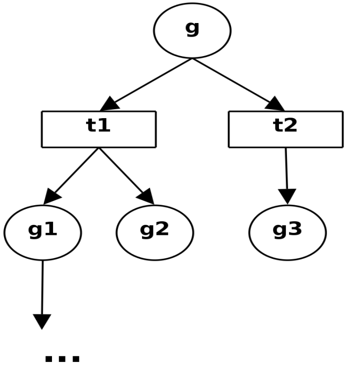
```
let rec first f l = function
  | [] → raise Failure
  | hd :: tl →
      try f hd
      with Failure → first f tl
and all gl (S, P) =
  match gl with
  | [] → S, P
  | g :: tl →
      let cl = cands (S, P) g in
      let S',P' = first (fun (S, P, gl) → all gl (S, P)) cl in
      all tl (S', P')
```

Choice points (tl in first) are kept by the OCaml stack but we need to show them to the user!

Making the stack explicit

- The status is $((P, gl, fl) :: alt, \theta)$
  - $\theta : Term \rightarrow Term + \bot$
  - $P = (\mathcal{P}, \Sigma)$
  - gl is the todo list
  - alt is the list of alternatives to the original problem
  - fl is the list of goals that fail if that item fails
- Goals in gl are of the form $D_g \mid S_g^t$ where g has an entry in $\mathcal{P}$

Making the stack explicit



- The initial status is $([P, [D_g], []], \emptyset)$
- If lemmas t_1 and t_2 apply to g, the new status will be $([P', [D_{g_1}; D_{g_2}; S_g^{t_1}], []\ ;\ P'', [D_{g_3}; S_g^{t_2}], [g]], \emptyset)$

The tactic I

$$(((\mathcal{P},\Sigma)\text{ as }P, S^t_g :: tl, fl) :: el, \theta) \xrightarrow{step} ((P, tl, fl) :: el', \theta') \qquad \text{(i)}$$
$$\text{when } \mathcal{M}(T) = \emptyset \text{ and } \Gamma \vdash ?g : T \in \mathcal{P}$$
$$\text{where } \theta' = \theta[T \mapsto \Sigma(g)] \text{ and } el' = purge(el, tl)$$

$$(((\mathcal{P},\Sigma)\text{ as }P, S^t_g :: tl, fl) :: el, \theta) \xrightarrow{step} ((P, tl, fl) :: el, \theta) \qquad \text{(ii)}$$
$$\text{when } \mathcal{M}(T) \neq \emptyset \text{ and } \Gamma \vdash ?g : T \in \mathcal{P}$$

$$(((\mathcal{P},\Sigma), D_g :: tl, fl) :: el, \theta) \xrightarrow{step} (((\mathcal{P},\Sigma'), tl, fl) :: el, \theta) \qquad \text{(iii)}$$
$$\text{when } \theta(T) \neq \bot \text{ and } \Gamma \vdash ?g : T \in \mathcal{P}$$
$$\text{where } \Sigma' = \Sigma \circ [?g := \theta(T)]$$

$$(((\mathcal{P},\Sigma), D_g :: tl, fl) :: el, \theta) \xrightarrow{step} (el, \theta'_{m+1}) \qquad \text{(iv)}$$
$$\text{when } \theta(T) = \bot \text{ and } \Gamma \vdash ?g : T \in \mathcal{P}$$
$$\text{where } \theta'_1 = \theta \text{ and } fl = \{g_1; \ldots ; g_m\}$$
$$\text{and } \Gamma_g \vdash ?g : T_g \in \mathcal{P} \text{ for } g \in \{1, \ldots, m\}$$
$$\text{and } \theta'_{g+1} = \theta'_g[T_g \mapsto \bot] \text{ for } g \in \{1, \ldots, m\}$$

The tactic II

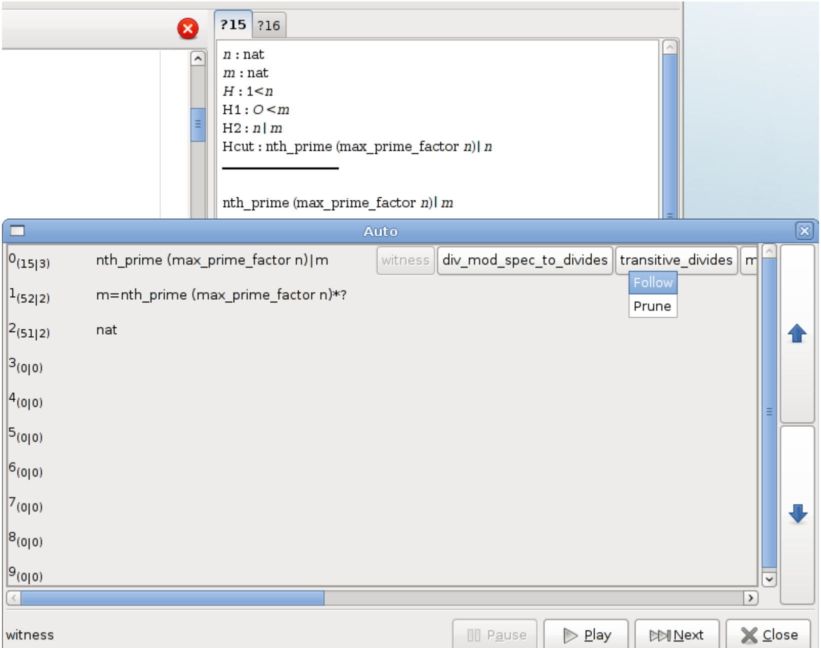
$$(((\mathcal{P}, \Sigma), D_g :: tl, fl) :: el, \theta) \xrightarrow{step} (el, \theta'_{m+1}) \qquad \text{(v)}$$
$$\text{when } cands(P, g) = []$$
$$\text{where } \theta'_1 = \theta \text{ and } fl = \{g_1; \ldots; g_m\}$$
$$\text{and } \Gamma_g \vdash ?g : T_g \in \mathcal{P} \text{ for } g \in \{1, \ldots, m\}$$
$$\text{and } \theta'_{g+1} = \theta'_g[T_g \mapsto \bot] \text{ for } g \in \{1, \ldots, m\}$$

$$((P, D_g :: tl, fl) :: el, \theta) \xrightarrow{step} ((P'_1, l_1 @ tl, []) :: \ldots \qquad \text{(vi)}$$
$$\ldots :: (P'_m, l_m @ tl, g :: fl) :: el, \theta)$$
$$\text{where } cands(P, g) = (t_1, P'_1, g_{1,1} \ldots g_{1,n_i}) :: \ldots$$
$$\ldots :: (t_m, P'_m, g_{m,1} :: \ldots :: g_{m,n_m})$$
$$\text{and } l_i = \mathcal{R}([D_{g_{i,1}} \ldots; D_{g_{i,n_i}}]) \circ [S_g^{t_i}] \text{ for } i \in \{1 \ldots m\}$$

$$((P, [S_g^t], fl) :: el, \theta) \xrightarrow{step} (\text{Success P}) \qquad \text{(vii)}$$

$$([], \theta) \xrightarrow{step} \text{Failure} \qquad \text{(viii)}$$

The interface



Conclusion

- We developed a goal-directed depth-first proof search procedure that can be driven by the user at runtime
- The interface helped in debugging the procedure

What's next?

- We will introduce Matita to first year students (logic course)
- Script reconstruction almost finished but needs some tuning

That's all

Thanks for your attention

If you want to give Matita a try:

http://matita.cs.unibo.it/