A Framework for Building Parallel ATPs

Geoff Sutcliffe and Kalvinder Singh
James Cook University

1 Introduction

Automated Theorem Proving (ATP) systems attempt to prove proposed theorems from given sets of axioms. The one major problem with ATP systems is that proving difficult theorems requires searching extremely large search spaces. This makes the use of ATP systems impractical in most situations, due to the too high resource requirements. Parallel ATP systems, such as ROO, PARROT, HPDS, METEOR, RCTHEO, and SPTHEO, have tackled this problem by harnessing more computing power and using proof-search strategies that have no obvious corresponding sequential strategies. Parallel ATP systems are successful in that they are often many times more effective (in terms of resource requirements) than sequential ATP systems. Most parallel ATP systems show good to very good processor utilization, when compared to other parallel and distributed applications.

Parallel ATP systems are often based on existing sequential ATP systems. For example, ROO is based on OTTER, and SPTHEO is based on SETHEO. This approach allows the sophistication of the underlying sequential ATP systems to be inherited by the parallel ATP systems. The underlying sequential ATP systems have inevitably been designed and written to run on workstations, and the corresponding parallel ATP systems are designed to (or are capable of) run on networks of such workstations. Further, some parallel ATP systems, e.g. RCTHEO, have been explicitly designed to run on networks of workstations. These types of parallel ATP systems exploit a coarse-grain approach to parallelism. A primary benefit of this parallelization strategy is the ready availability of the 'parallel hardware'.

Considering that parallel ATP systems improve the efficacy of theorem proving and that coarse grained parallel hardware is readily available in the form of workstation networks, parallel ATP systems are not as widely developed and used as they could be. This is presumably because parallel ATP systems are significantly harder to design, implement, and use.

- At the design level, an appropriate (in senses such as completeness, fairness, etc) parallel ATP algorithm has to be developed, which in itself is a difficult task. Issues of intercomponent communication and synchronization have to be addressed. It can be the case that the design will be the work of a logician (as opposed to a computer scientist) who does not have the requisite skills to translate the design into a operational parallel ATP system.
- At the implementation level, the implementor often has to cope with low level communication primitives. Such primitives may or may not work harmoniously with the implementation language. The implementation will often capture the system configuration in code, and reconfiguration requires significant effort. As a result experimentation with different designs is limited.
- At the user level, it is hard to monitor and interact with the execution of a parallel ATP system. A technical reason for this is that the component processes often do not have user I/O streams. A more human reason is that parallel ATP systems are often designed and implemented by computer scientists for computer scientists, and the user interface is cryptic.

The need is for a framework within which parallel ATP system designs can easily be translated into working systems. The framework must allow existing sequential ATP systems to

be combined with little or no modification. Interprocess communication and synchronization must be simple, and implemented in terms of well known programming constructs. The framework must provide an easy to use interface to allow the user (who may be only semi-computer literate) to monitor and control all aspects of the parallel system. The X-windows Parallel ATP (XPATP) framework meets these requirements.

XPATP is a graphical user interface for building parallel ATP systems out of component sequential ATP systems, to run on networks of machines connected on the InterNet. This particular type of Parallel ATP system is called a *PATP system*, and the component sequential ATP systems are called *PATP components*. Intercomponent communication and synchronization is implemented in terms of the standard input and output streams of each PATP component, so that this facet is easily codable in the PATP components. The user interacts with XPATP using a point-and-click mouse interface, so very little keyboard interaction is required. The interface makes it easy to specify the PATP system's architecture, in terms of the PATP components and the required communication links. The user is able to monitor and control the execution of the PATP system in terms of the PATP components and the communication links.

2 The Basics

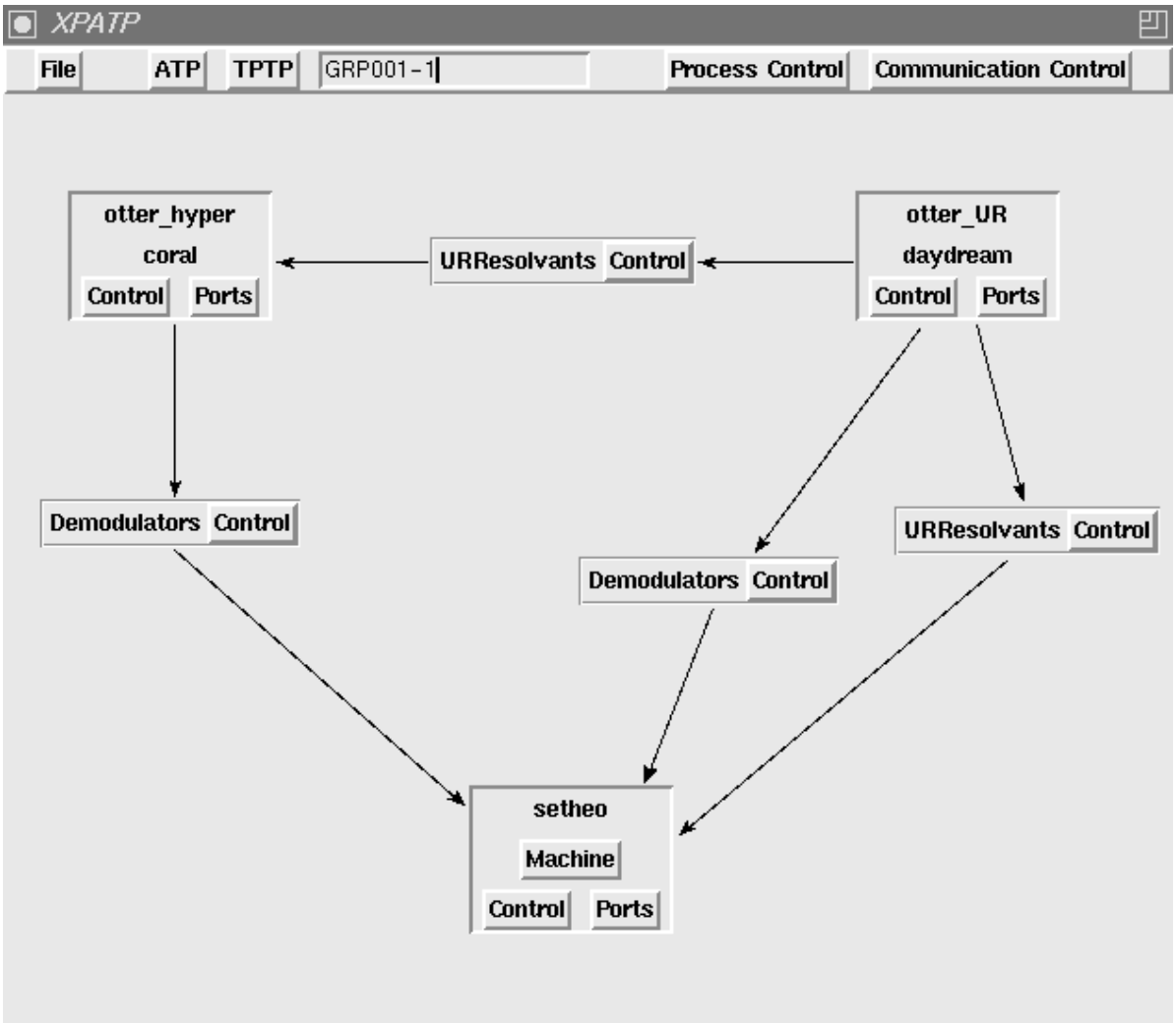

Figure 1: An Example XPATP Window

The execution of XPATP will cause a window to appear on the screen. Figure 1 shows a sample window. The window has two sections, the menu bar and the work area. The menu bar has five menus and a display area. Their uses are described in Sections 3 and 5. The work area is used to display the configuration of the PATP system. Each PATP component is represented by an *ATP frame*, in which the component name and host machine name are given. Each ATP frame also has two menus, one for controlling the execution of the component and one that lists the *output ports* of that component. Each output port can be attached to multiple simplex communication links that lead to other PATP components. Any data or control messages written to an output port are placed onto all the associated communication links. The communication links are represented in the work area by *link lines* between the corresponding ATP frames. Link lines have arrow heads indicating the direction of flow, and are broken by *link frames*. Each link frames contains the name of the output port from which the communication link receives messages, and a menu for controlling the communication link.

3 Configuration

The available PATP components and host machines are specified by the user, in configuration files.

The `XPATPComponents` file lists the available PATP components, using the syntax `<ATP Name>:<Command>:<Output port name>,...,<Output port name>`. For example,

```
otter_hyper:~geoff\ATPSystems\otter <%s.hyper_in:Demodulators
otter_UR:~geoff\ATPSystems\otter <%s.ur_in:URResolvants,Demodulators
setheo:~geoff\ATPSystems\setheo %s:Lemmas
```

The `<ATP Name>` field contains the name of the PATP component. The `<Command>` field is the command line instruction that invokes the PATP component. This field can contain `%s` parts. Before the command line is executed, each occurrence of a `%s` is replaced by the user specified string (typically the name of the input file containing the theorem to be proved) in the menu display area. The `<Output port name>` field lists the output ports of that PATP component.

The `XPATPMachines` file lists the names of the machines that XPATP can use, using the syntax `<Machine Name>:<InterNet Name>`. For example,

```
coral:coral.cs.jcu.edu.au
daydream:daydream.cs.jcu.edu.au
sailfish:sailfish.jcu.edu.au
```

Configuration of a PATP system is done interactively using the menus and the work area. Initially the work area is empty. PATP components are selected from the **ATP** menu, which lists the PATP components given in the `XPATPComponents` file. The selection of a PATP component from the menu causes a movable ATP frame to appear in the work area. The frame's **Machine** menu button is used to select what machine the component is to execute on. The **Machine** menu lists the machines given in the `XPATPMachines` file.

A communication link between two PATP components is formed by selecting an output port from the **Ports** menu of the source component's ATP frame, and dragging to the destination component's ATP frame. The **Ports** menu lists the output ports specified for that PATP component in the `XPATPComponents` file.

The **File** menu contains options to save and load configurations, allowing configured PATP systems to be reused.

4 Execution of a PATP System

To specify the theorem to be attempted by a configured PATP system, the display area in the menu bar needs to contain a value. This value is used to replace `%s` parts of the command lines supplied in the `XPATPComponents` file. The value may be obtained using the **TPTP** menu, which allows the user to browse the TPTP Problem Library (if available, which it should be!) for a theorem to attempt. Otherwise, the display area can be entered manually.

When all is ready, the PATP system is executed by selecting the **Start** option in the **Process Control** menu. This causes all the PATP components to be executed on the specified machines, by executing the specified command lines (after replacement of `%s` parts). The PATP components execute as normal, with the exception of their standard output. XPATP filters the stdout streams of the PATP components for lines with the formats `:::<Port>:<Message>` and `:::<Port>!<Message>`. Such output lines are used to implement intercomponent communication and synchronization. *Data messages*, of the form `:::<Port>:<Message>`, are copied onto all the communication links connected to the `<Port>`. *Control messages*, of the form `:::<Port>!<Message>`, are intercepted and interpreted by XPATP. The possible values for `<Message>` in control messages are `start`, `kill`, `suspend`, `resume`, and `interrupt(N)`. The effect of these messages on the recipient components is described in Section 5. As well as the output ports listed in the `XPATPComponents` file, there are two special `<Port>` values, `all` and `self`, which can be used by an XPATP component. If the `<Port>` value is `all` then the `<Message>` is destined for all output ports of the PATP component. If the `<Port>` value is `self` then the `<Message>` is destined for the sending PATP component itself.

Any standard output lines that are not filtered out are put in a stdout window for that ATP frame, as described in Section 6.

5 System Control

As indicated in Section 4, overall control of a PATP system is exercised through the **Process Control** menu. The process control options are **Start**, **Stop**, **Suspend**, **Resume**, **Interrupt(N)**, **View**, and **Remove**. These menu options are duplicated at the individual component level in the **Control** menus of the ATP frames, and are also used in intercomponent control messages as described in Section 4. The effects of the process control options are:

- Start: Start the PATP components(s), on the specified machine(s), by executing the command line specified.
- Stop: Stop the PATP components(s). This kills the ATP component'(s') process(es).
- Suspend: Suspend the PATP components(s). If a component is already suspended then this has no effect.
- Resume: Resume execution of the suspended PATP components(s). If a component is not suspended then this has no effect.
- Interrupt(N): Send interrupt number N to the PATP components(s). The Stop option is implemented in terms of this option, with N = 9.
- View: Creates a display window(s) containing the genuine standard output from the PATP component(s).
- Remove: Removes the PATP component(s) from the PATP system. Any communication links attached to the PATP component(s) are also removed.

In combination with the special <Port> values, control messages facilitate neat control of PATP components. For example, if one PATP component needs to synchronize with another, it can send a `suspend` control message to itself, and the second PATP component sends a `resume` message when it is ready.

Overall control of the communication links is exercised through the **Communication Control** menu. The communication control options are **Clear**, **Suspend**, **Resume**, **View**, and **Remove**. These menu options are duplicated at the individual link level in the **Control** menus of the link frames. The effects of the control options are:

- Clear: Remove all messages from the communication link(s).
- Suspend: Stop placing messages on the communication link(s). Messages written to the output port(s) are buffered.
- Resume: Resume placing messages on the communication link(s). Any buffered messages are placed on the communication link(s) first.
- View: Creates a display window(s) containing messages on the communication link(s).
- Remove: Removes the communication link(s) from the PATP system. Any messages on the communication link(s) are discarded.

6 System Monitoring

All genuine standard output from PATP components and all intercomponent messages can be monitored in separate display windows. Genuine standard output from PATP components is viewed in scrollable *stdout windows*. Intercomponent messages are viewed in scrollable *message windows*. All windows are accessed through the **View** option in the **Control** menus of the ATP frames and link frames, or from the main **Process Control** and **Communication Control** menus.

Control of display windows is exercised through the option bar at the top of the window. The options are **Quit**, **Pause**, **Add**, and **Delete**. The effects of the control options are:

- Pause: Stops the display of any new output. To resume the display of output click on the pause option again.
- Add: Add a user specified message to the communication link. This option only appears in message windows.
- Delete: Remove an indicated message from the communication link. This option only appears in message windows.
- Quit: Removes the window from the screen.

7 Conclusion

XPATP is being implemented in tcl/tk. Tcl/tk is a simple scripting language which is freely distributed from a number of sites around the world. The language is specifically designed to build user interfaces. XPATP fully utilises the facilities of tcl/tk, to make an easy to use interface for designing and implementing PATP systems.