A Short Introduction To Using KeYmaera

Yanni Kouskoulas

17 February, 2012

1 Overview

KeYmaera is a tool that mechanises a subset of differential dynamic logic. It allows one to write hybrid programs, make logical assertions about these programs, and then prove or disprove those assertions.

KeYmaera is built on the Key prover, which was designed to be used to prove properties about Java programs. It also uses a number of other solving tools, including Mathematica, to do back end manipulation of its goals and assumptions. These solving tools, as well as the proof rules that are implemented for differential dynamic logic, must be proved sound and implemented correctly; the Key prover does not otherwise guarantee the soundness of the transformations it makes. I.e. if Mathematica has a bug, KeYmaera's soundness may suffer.

2 Proving Using KeYmaera

Interacting with KeYmaera involves interleaving the automated proof search, and manual application of tactics. The automated proof search starts by clicking the "Start" button, and ends continues until it stops by itself and asks the user for guidance, or the user stops it by clicking the stop button. The manual application of tactics requires the user to examine the proof state, and then, by right-clicking on proof terms or operators in the logic-state pane, selecting an appropriate tactic to transform either the goals or assumptions in a way that allows the proof to progress. In many cases, manual application of a tactic to a term in the proof will bring up a dialog box in which the user can enter further information about the specifics of the tactic and how it should be applied. The primary way to guide KeYmaera manually is to apply the "cut" proof rule, which introduces a lemma as an assumption, after proving it using the current proof state. Other tactics allow one to manipulate universal quantifiers and implications and other terms on either side of the sequent.

The language in which the hybrid system is described in many situations allows the system to automatically apply rules that decompose the system into its consituent parts. Once the KeYmaera stops or is stopped manually, the user can apply one or more of its tactics to nudge it in a particular direction, and then set it off again to search automatically from the modified proof state. Stylistically, this is somewhat similar to ACL2, with the prover taking away detailed control in order to automate most of the tedious manipulation.

3 Interacting with KeYmaera's User Interface

KeYmaera has a simple, intuitive graphical user interface that consists of three panes, a message area across the bottom of the window, and a button bar at the top of the window. KeYmaera represents its hybrid programs and assertions about them in a file with the suffix .key, which borrows syntax from the Key prover. The main use of the button bar is to load a .key file, and to start and stop the automatic proof search. The fourth button from the left, a green folder, will bring up a dialog box that allows you to select a .key file to load. Once a .key file has been loaded, the theorem to be proven is displayed in the right pane of the window (the logic-state pane, discussed below). Pressing the "Start" button begins the automated proof search.

3.1 Proof Pane

The *Proof pane* has three tabs, and takes up the lower left quadrant of the screen. The *Proof* tab, the *Hybrid strategy* tab, and the *Goals* tab are all available. The *Hybrid strategy* tab shows KeYmaera's different settings that determine what back-end mathematics packages and libraries it uses for different tasks. The *Goals* tab has a list of current goals to be proven.

The most important tab in this pane is the *Proof* tab, which displays the proof steps, so far, with indentation showing the different cases. Some of these proof steps are complicated tactics over which the user has limited control, e.g. “ODE Solve” or “Eliminate Universal Quantifiers”, tactics that depend on Mathematica or another back end to solve a differential equation or manipulate a term by eliminating universal quantifiers and deciding the truth of the goal. Other proof steps are more predictable in their outcome, e.g. the cut tactic, that allows a user to insert a specific assumption into the proof state. Different cases in the proof can be expanded or hidden by clicking on the box to the left of the tactic that generated them; it will have a [+] or [–] to indicate whether it has been expanded. Branches of the proof whose obligations have been discharged are colored green in this view. Branches that are yet to be done, are colored red. Clicking on a proof step highlights that line of the proof, and then displays the proof state at that point in the *Logic State* pane to the right.

3.2 Logic State Pane

The *Logic State* pane, on the right-hand side of the window, shows the proof state at a single point in time. This pane displays a comma-separated set of assumptions, followed by a token `==>`, followed by a comma-separated set of goals (logical assertions) we wish to prove. One can think of this as a sequent, with the terms to the left of the `==>` being the antecedent set, and those following it being the succedent set of formulas. The logical assertion to be proven in many cases also contains a hybrid program, which is also displayed in this window. As the proof progresses, this sequent undergoes a number of transformations to show the state of the proof at any given point in time. When KeYmaera gets “stuck” and needs user guidance, this is the pane that the user needs to look at to determine what obligations need to be proved and what assumptions are available.

Using the mouse to hover over a term in the *Logic State* pane highlights the term, so one can see the extent of that term according to the precedence of operators in an expression. Thus, this functionality automatically provides paren-matching-equivalent functionality.

Right clicking on an operator or term in this window opens a pop-up menu that has the various tactics available for manipulating that term. If one is selected and more information is needed, a dialog box opens in which the appropriate information can be entered. In any case, when a tactic is manually chosen to manipulate the proof state, its name is entered into the list of steps in the *Logic State* pane, with a small rightward pointing hand to indicate that it was chosen manually. When the user is done manually choosing tactics, he or she can press start to continue the automated proof search from the current state.

3.3 Task Pane

The *Task* pane has a list of different theorems that are to be proven. Clicking on a file name switches the other two panes to the state of that proof and its current goal. Right clicking on a task in the *Task* pane pops up a menu that allows you to abandon the task, removing it from the task list.

4 Tactics

This section contains a sampling of commonly used tactics, so the reader can get a feeling of how they operate and how they are named. The tactics KeYmaera lists in the pop-up menu are context dependent; it only lists tactics that can usefully be applied in that place, so not all tactics are available at all times. This listing is incomplete, but gives the user a starting point to help guide KeYmaera.

Tactic Name	Initial Proof State	After Tactic
induction_loop	$\Gamma \vdash [\alpha*]\phi$	$\Gamma \vdash \psi$
		$\Gamma \vdash \psi \to [\alpha]\psi$
		$\psi \vdash \phi$
all_right	$\Gamma \vdash \forall x, G$	$\Gamma, x \vdash G$
all_left	$\forall x, \Gamma \vdash G$	$\Gamma_x^{x'} \vdash G$
imply_right	$\Gamma \vdash A \to B$	$\Gamma, A \vdash B$
imply_left	$\Gamma, A \to B \vdash G$	$\Gamma \vdash A$
		$\Gamma, B \vdash G$
or_left	$\Gamma, A \vee B \vdash G$	$\Gamma, A \vdash G$
		$\Gamma, B \vdash G$
and_left	$\Gamma, A \wedge B \vdash G$	$\Gamma, A, B \vdash G$
axiom_close	$\Gamma \vdash$ true	
hide_left	$\Gamma, A \vdash G$	$\Gamma \vdash G$
hide_right	$\Gamma \vdash G, A$	$\Gamma \vdash G$
cut	$\Gamma \vdash G$	$\Gamma, C \vdash G$
		$\Gamma \vdash C$

The naming scheme is intended to be descriptive of the tactic's function:

the first part of the name (e.g. all, imply) refers to the outermost operator in the term being transformed, and the second part of the name (i.e. left, right) refers to which side of the sequent the tactic is designed to be applied.

5 Useful advice for beginning KeYmaera users

- If you manually stop KeYmaera and it is in the middle of the *Eliminate Universal Quantifiers* tactic it will produce the message "An exception occurred during strategy execution. java.lang.IllegalStateException: Calculation aborted!" In this situation, you should help KeYmaera along my manipulating the goal to simplify the proof state before continuing the automatic search.

- Move constraints from the precondition into the enclosing hybrid program, because KeYmaera handles constraints in the precondition less efficiently than assignments in the hybrid program. Too many constraints in the precondition, even if they are equalities, can make KeYmaera unresponsive. This means that you should convert programs that look like this:

  ```
  \[ R a \] (
  (pre&(a=0)) -> \[prog\](post))
  ```

  into this

  ```
  \[R a ; a := 0 \] (
  (pre) -> \[prog\](post))
  ```