The Key-Project

- Theorem Prover
- Developed at University of Karlsruhe
- [http://www.key-project.org/](http://www.key-project.org/)
- Theory specialized for Java(Card).
- Can generate proof-obligations from JML specification.
- Underlying theory: Sequent Calculus + Dynamic Logic
Dynamic logic extends predicate logic by

- $[\alpha] \phi$
- $\langle \alpha \rangle \phi$

where $\alpha$ is a program and $\phi$ a sub-formula.

The meaning is as follows:

- $[\alpha] \phi$: after all terminating runs of program $\alpha$ formula $\phi$ holds.
- $\langle \alpha \rangle \phi$: after some terminating run of program $\alpha$ formula $\phi$ holds.
Comparison with Hoare Logic

The sequent $\phi \implies [\alpha] \psi$ corresponds to partial correctness of the Hoare formula:

$$\{\phi\} \alpha \{\psi\}$$

If $\alpha$ is deterministic, $\phi \implies \langle \alpha \rangle \psi$ corresponds to total correctness.
How can we use equivalences in Sequent Calculus?

Add the rule $\Gamma[\psi/\phi] \implies \Delta[\psi/\phi]$, where $\phi \equiv \psi$.

This is similar to applyEq.
Dynamic Logic is Modal Logic

- $\langle \alpha \rangle \phi \equiv \neg [\alpha] \neg \phi$
- $[\alpha] \phi \equiv \neg \langle \alpha \rangle \neg \phi$

Furthermore:
- if $\phi$ is a tautology, so is $[\alpha] \phi$
- $[\alpha] (\phi \rightarrow \psi) \rightarrow ([\alpha] \phi \rightarrow [\alpha] \psi)$

Remark: For deterministic programs also the reverse holds

$([\alpha] \phi \rightarrow [\alpha] \psi) \rightarrow [\alpha] (\phi \rightarrow \psi)$
How can we express that program $\alpha$ must terminate?

$$\langle \alpha \rangle \text{true}$$

This can be used to relate $[\alpha]$ and $\langle \alpha \rangle$:

$$\langle \alpha \rangle \phi \equiv [\alpha] \phi \land \langle \alpha \rangle \text{true}$$
KeY distinguishes the following symbols:

- **Rigid Functions**: These are functions that do not depend on the current state of the program.
  
  - \(+, -, \times : \text{integer} \times \text{integers} \rightarrow \text{integer}\) (mathematical operations)
  - \(0, 1, \ldots : \text{integer}, \text{TRUE}, \text{FALSE} : \text{boolean}\) (mathematical constants)

- **Non-Rigid Functions**: These are functions that depend on current state.
  
  - \([. ] : \top \times \text{int} \rightarrow \top\) (array access)
  - \(\text{.next} : \top \rightarrow \top\) if \text{next} is a field of a class.
  - \(i, j : \top\) if \(i, j\) are program variables.

- **Variables**: These are logical variables that can be quantified. Variables may not appear in programs.
  
  - \(x, y, z\)
Example

\[ \forall x. i = x \rightarrow \langle \{ while(i > 0) \{ i = i - 1; \} \} \rangle i = 0 \]

- 0, 1, – are rigid functions.
- > is a rigid relation.
- \( i \) is a non-rigid function.
- \( x \) is a logical variable.

Quantification over \( i \) is not allowed and \( x \) must not appear in a program.