The Calculus LKS and Handy LKS Language

Cvetan Dunchev, Mikheil Rukhaia

Institute of Computer Languages, Vienna University of Technology.

Abstract. This is a draft paper, describing the calculus **LKS** and its machine readable prototype, called **HLKS**.

1 The Calculus LKS

In this section we briefly describe an implementation of the schematic sequent calculus in the GAPT Framework.[1] We implemented the schematic propositional language in general format. It can be extended also to the first-order language easily. Our schematic sequent calculus **LKS** uses usual propositional **LK** rules, which was already implemented and additionally some equivalence rules to derive the necessary main formulas of the inferences. We start with some basic definitions.

Definition 1.1 (Indexed proposition) *An expression of the form P_a, where a is a linear arithmetic expression built over the signature $0, s, +$ and integer variables, is called an indexed proposition. If a does not contain integer variables then we speak about* ground *indexed propositions, which are called* propositional variables. *Integer variables can be free or bound. Free integer variables are called parameters.*

Definition 1.2 (Formula schemata) *We define formula schemata inductively in the following way:*

- *An indexed proposition is an (atom) formula schema.*
- *If ϕ_1 and ϕ_2 are formula schemata, then so are $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$ and $\neg\phi_1$.*
- *If ϕ is a formula schema, a, b are arithmetic expressions and i is an index variable not bound in ϕ, then $\bigwedge_{i=a}^{b} \phi$ and $\bigvee_{i=a}^{b} \phi$ are formula schemata, called iterations (i becomes bound under the iterations).*

Definition 1.3 (Sequent schemata) *An expression of the form $\Gamma \vdash \Delta$, where Γ and Δ are multisets of formula schemata, is called a sequent schema. Γ is called antecedent and Δ a succeedent of the sequent. If $\Gamma = \Delta = \{A\}$, for A being an indexed proposition, then it is called an initial sequent schema.*

Definition 1.4 (Sequent Context) *We say that $C[A]$ is a sequent context, if $C[A]$ is a sequent which contains A either in its antecedent, or succeedent.*

[1] Home page: http://code.google.com/p/gapt/

Definition 1.5 (Proof links) *An expression of the form* $\frac{(\varphi, a)}{S}$ (dashed line), *where* φ *is a proof name,* a *is an arithmetic expression and* S *is an end-sequent of* φ *at iteration* a*, is called a* proof link.

Definition 1.6 (Substitution) *A substitution is a function mapping every (free) integer variable to an arithmetic expression.*

Definition 1.7 (Calculus LKS) *Our sequent calculus* **LKS** *contains initial sequent schemata or proof links as axioms and consists of the following rules:*

1. *Logical rules:*
   - $\neg$ *introduction*

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}\ \neg\colon l \qquad \text{and} \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\ \neg\colon r$$

   - $\wedge$ *introduction*

$$\frac{A, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ \wedge\colon l1 \qquad \text{and} \qquad \frac{B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ \wedge\colon l2$$

$$\frac{\Gamma \vdash \Delta, A \qquad \Pi \vdash \Lambda, B}{\Gamma, \Pi \vdash \Delta, \Lambda, A \wedge B}\ \wedge\colon r$$

   Note that $A(0) \equiv \bigwedge_{i=0}^{0} A(i)$ *and* $(\bigwedge_{i=0}^{n} A(i)) \wedge A(n+1) \equiv \bigwedge_{i=0}^{n+1} A(i)$*. Below we will describe corresponding equivalence rules.*
   - $\vee$ *introduction*

$$\frac{A, \Gamma \vdash \Delta \qquad B, \Pi \vdash \Lambda}{A \vee B, \Gamma, \Pi \vdash \Delta, \Lambda}\ \vee\colon l$$

$$\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B}\ \vee\colon r1 \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B}\ \vee\colon r2$$

   Note that $A(0) \equiv \bigvee_{i=0}^{0} A(i)$ *and* $(\bigvee_{i=0}^{n} A(i)) \vee A(n+1) \equiv \bigvee_{i=0}^{n+1} A(i)$*. Below we will describe corresponding equivalence rules.*
2. *Structural rules:*
   - *Weakening rules:*

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ w\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ w\colon r$$

   - *Contraction rules:*

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ c\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ c\colon r$$

   - *Cut rule:*

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Pi \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda}\ cut$$

3. *Some "schortcuts" and Equivalence rules:*
   - $\wedge$ *introduction left*

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ \wedge\colon l$$

   is shortcut for

$$\cfrac{\cfrac{\cfrac{A, B, \Gamma \vdash \Delta}{A \wedge B, B, \Gamma \vdash \Delta}\ \wedge\colon l1}{A \wedge B, A \wedge B, \Gamma \vdash \Delta}\ \wedge\colon l2}{A \wedge B, \Gamma \vdash \Delta}\ c\colon l$$

   - $\vee$ *introduction right*

$$\frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B} \vee\colon r$$

is shortcut for

$$\cfrac{\cfrac{\cfrac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B, B}\ \vee\colon r1}{\Gamma \vdash \Delta, A \vee B, A \vee B}\ \vee\colon r2}{\Gamma \vdash \Delta, A \vee B}\ c\colon r$$

– $\wedge$ *equivalence rules:*

$$\frac{C[(\bigwedge_{i=a}^{b} A_i) \wedge A_{b+1}]}{C[(\bigwedge_{i=a}^{b+1} A_i)]} \equiv\colon \wedge 1$$

$$\frac{C[(\bigwedge_{i=a+1}^{b} A_i) \wedge A_a]}{C[(\bigwedge_{i=a}^{b} A_i)]} \equiv\colon \wedge 2$$

$$\frac{C[A_a]}{C[\bigwedge_{i=a}^{a} A_i]} \equiv\colon \wedge 3$$

– $\vee$ *equivalence rules:*

$$\frac{C[(\bigvee_{i=a}^{b} A_i) \vee A_{b+1}]}{C[(\bigvee_{i=a}^{b+1} A_i)]} \equiv\colon \vee 1$$

$$\frac{C[(\bigvee_{i=a+1}^{b} A_i) \vee A_a]}{C[(\bigvee_{i=a}^{b} A_i)]} \equiv\colon \vee 2$$

$$\frac{C[A_a]}{C[\bigvee_{i=a}^{a} A_i]} \equiv\colon \vee 3$$

An **LKS**-proof is called *ground* if it does not contain free parameters, index variables, or proof links.

Definition 1.8 (Proof schemata) *Let $\psi^1, \ldots, \psi^m$ be proof symbols and $S^1, \ldots, S^m$ be sequents containing the free parameter n. Then, a* proof schema Ψ *is a tuple of pairs*

$$\left\langle (\psi^1_{\text{base}}, \psi^1_{\text{step}}), \ldots, (\psi^m_{\text{base}}, \psi^m_{\text{step}}) \right\rangle$$

such that:

1. *ψ^i_{base} is a ground* **LKS***-proof of $S^i\,\{n \leftarrow 0\}$, for all $i = 1, \ldots, m$,*
2. *ψ^i_{step} is an* **LKS***-proof of $S^i\,\{n \leftarrow k+1\}$, where k is a parameter of ψ^i_{step}, and ψ^i_{step} contains only proof links of the form:*

$$\frac{(\psi^i, k)}{S^i\,\{n \leftarrow k\}} \qquad \text{and/or} \qquad \frac{(\psi^j, a)}{S^j\,\{n \leftarrow a\}}$$

where $i < j$ and a is an arithmetic expression.

We assume an identification between formula occurrences in the end-sequents of $\psi^i_{\text{base}}, \psi^i_{\text{step}}$ (so that we can speak of occurrences in the end-sequents of ψ^i). We also say that S^1 is the end-sequent of Ψ.

2 The Language HLKS

In this section we describe the Handy **LKS** language in left-linear grammar. We use the following conventions: an expression $[\ldots]$ is used to denote the optional part of a definition, but expressions $[\ldots]^*$ or $[\ldots]^+$ has the standard notion of a regular expression. In the first case we have zero or more repetitions. In the second case - at least one repetition. Also we use $0-9$, $a-z$ and $A-Z$ expressions, to denote the range of digits, lowercase letters and uppercase letters respectively. Braces such as (,), { and } are part of the syntax and omitting them will throw an exception. **LK**-proofs can also be specified using this grammar. For this reason the *step* block of a proof definition should be empty, i.e. only the *base* block of a proof definition will be used. Finally, the **HLKS**-parser which parses this grammar is not sensitive to white spaces and new lines.

$$
\begin{array}{rcl}
\langle lks\_file \rangle & ::= & [\langle lks\_statement \rangle]^* \\
\langle lks\_statement \rangle & ::= & \langle definition \rangle \\
 & | & \langle proof \rangle \\
\langle definition \rangle & ::= & \langle formula \rangle := \langle formula \rangle \\
\langle proof \rangle & ::= & \text{proof } \langle proof\_name \rangle \text{ proves } \langle sequent \rangle \\
 & & \text{base } \{ \ \langle inference\_list \rangle \ \} \\
 & & \text{step } \{ \ \langle inference\_list \rangle \ \} \\
\langle proof\_name \rangle & ::= & [\backslash][a-z, 0-9]^+ \\
\langle sequent \rangle & ::= & [\langle formula\_list \rangle] \mid - \ [\langle formula\_list \rangle] \\
\langle formula\_list \rangle & ::= & \langle formula \rangle \\
 & | & \langle formula \rangle, \langle formula\_list \rangle \\
\langle inference\_list \rangle & ::= & [\langle inference \rangle]^+ \\
\langle inference \rangle & ::= & \langle id \rangle : \langle rule \rangle \\
 & | & \text{root}: \langle rule \rangle \\
\langle id \rangle & ::= & [0-9, a-z]^+ \\
\langle int\_var \rangle & ::= & [i, j, k, l, m, n]^+ [0-9]^* \\
\langle int\_const \rangle & ::= & [0-9]^+ \\
\langle predicate\_name \rangle & ::= & [A-Z]^+ [a-z, 0-9]^* \\
\langle indexed\_predicate \rangle & ::= & \langle predicate\_name \rangle (\langle arithm\_expr\_list \rangle) \\
\langle formula \rangle & ::= & \langle indexed\_predicate \rangle \\
 & | & \sim \langle formula \rangle \\
 & | & (\langle formula \rangle \ /\backslash \ \langle formula \rangle) \\
 & | & (\langle formula \rangle \ \backslash/ \ \langle formula \rangle) \\
 & | & \langle iteration \rangle \langle formula \rangle \\
\langle iteration \rangle & ::= & \langle iter\_symbol \rangle (\langle int\_var \rangle = \langle arithm\_expr \rangle .. \langle arithm\_expr \rangle)
\end{array}
$$

$$
\begin{aligned}
\langle \mathit{iter\_symbol} \rangle &::= \mathrm{BigAnd} \\
&\mid\ \mathrm{BigOr} \\
\langle \mathit{arithm\_expr\_list} \rangle &::= \langle \mathit{arithm\_expr} \rangle \\
&\mid\ \langle \mathit{arithm\_expr} \rangle, \langle \mathit{arithm\_expr\_list} \rangle \\
\langle \mathit{arithm\_expr} \rangle &::= \langle \mathit{int\_var} \rangle \\
&\mid\ \langle \mathit{int\_const} \rangle \\
&\mid\ \langle \mathit{int\_var} \rangle + \langle \mathit{int\_const} \rangle \\
\langle \mathit{rule} \rangle &::= \mathrm{ax}(\langle \mathit{sequent} \rangle) \\
&\mid\ \mathrm{pLink}((\langle \mathit{proof\_name} \rangle, \langle \mathit{index} \rangle)\langle \mathit{sequent} \rangle) \\
&\mid\ \mathrm{negL}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{negR}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andL1}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andL2}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andL}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andR}(\langle \mathit{id} \rangle, \langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orL}(\langle \mathit{id} \rangle, \langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orR1}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orR2}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orR}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{weakL}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{weakR}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{contrL}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{contrR}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{cut}(\langle \mathit{id} \rangle, \langle \mathit{id} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andEqL1}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andEqR1}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andEqL2}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andEqR2}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andEqL3}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{andEqR3}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orEqL1}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orEqR1}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orEqL2}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orEqR2}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orEqL3}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle) \\
&\mid\ \mathrm{orEqR3}(\langle \mathit{id} \rangle, \langle \mathit{formula} \rangle, \langle \mathit{formula} \rangle)
\end{aligned}
$$

All the inferences, but the *axiom* and the *proof-link* has an *id* for unary-inferences and two *id*'s for binary inferences which are the corresponding upper sub-proofs. The $formula$(s) in the inferences are the auxiliary formula(s).

3 An Example

To better understand the calculus and grammar described above, we illustrate it with simple example. Let's consider the following proof schema $\Psi = \langle(\psi_{\text{base}}, \psi_{\text{step}})\rangle$ of a sequent $P_0, \bigwedge_{i=0}^{k}(\neg P_i \lor P_{i+1}) \vdash P_{k+1}$, where ψ_{base} is:

$$\dfrac{\dfrac{\dfrac{P_0 \vdash P_0}{\neg P_0, P_0 \vdash}\,\neg : l \qquad P_1 \vdash P_1}{P_0, \neg P_0 \lor P_1 \vdash P_1}\,\lor : l}{P_0, \bigwedge_{i=0}^{0} \neg P_i \lor P_{i+1} \vdash P_1}\,\equiv :\ \land 3$$

and ψ_{step} is:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\psi, k)}{P_0, \bigwedge_{i=0}^{k}(\neg P_i \lor P_{i+1}) \vdash P_{k+1}} \qquad \dfrac{\dfrac{P_{k+1} \vdash P_{k+1}}{\neg P_{k+1}, P_{k+1} \vdash}\,\neg : l \qquad P_{k+2} \vdash P_{k+2}}{P_{k+1}, \neg P_{k+1} \lor P_{k+2} \vdash P_{k+2}}\,\lor : l}{P_0, \bigwedge_{i=0}^{k}(\neg P_i \lor P_{i+1}), \neg P_{k+1} \lor P_{k+2} \vdash P_{k+2}}\,cut}{P_0, \bigwedge_{i=0}^{k}(\neg P_i \lor P_{i+1}) \land (\neg P_{k+1} \lor P_{k+2}) \vdash P_{k+2}}\,\land : l1, \land : l2, c : l}{P_0, \bigwedge_{i=0}^{k+1}(\neg P_i \lor P_{i+1}) \vdash P_{k+2}}\,\equiv :\ \land 1$$

Then this proof can be written in our grammar in the following way:

```
proof  \psi proves P(0), BigAnd(i=0..k) (~ P(i) \/ P(i+1)) |- P(k+1)
base {
        1: ax(P(0) |- P(0))
        2: negL(1, P(0))
        3: ax(P(1) |- P(1))
        4: orL(2, 3, ~ P(0), P(1))
        root: andEqL3(4, (~ P(0) \/ P(1)), BigAnd(i=0..0) (~ P(i) \/ P(i+1)))
}
step {
        1: pLink((\psi, k) P(0), BigAnd(i=0..k) (~ P(i) \/ P(i+1)) |- P(k+1))
        2: ax(P(k+1) |- P(k+1))
        3: negL(2, P(k+1))
        4: ax(P(k+2) |- P(k+2))
        5: orL(3, 4, ~ P(k+1), P(k+2))
        6: cut(1, 5, P(k+1))
        7: andL(6, BigAnd(i=0..k) (~ P(i) \/ P(i+1)), (~ P(k+1) \/ P(k+2)))
        root: andEqL1(7, (BigAnd(i=0..k) (~ P(i) \/ P(i+1)) /\ (~ P(k+1) \/ P(k+2))),
                BigAnd(i=0..k+1) (~ P(i) \/ P(i+1)))
}
```