Sequent Calculus

Robert Keller
February 2013
Sequent Calculus (SC)

• The **sequent calculus** was used by Gentzen (he called it “System S”) to derive results about Natural Deduction.

• SC allows **sets** of formulas on **both** sides of |–.

• In the SC, |– becomes an **object-language** symbol, rather than a meta-language symbol as we have been using it. However, some authors use ⇒ or → instead of |– for SC.

• It is easier to implement in software a proof generator based on SC than it is one based on ND.
Meaning of $\vdash$ in SC

- The intuitive meaning of

$$A_1, \ldots, A_m \vdash B_1, \ldots, B_n$$

“Using hypotheses in the set \{A_1, \ldots, A_m\}
one can derive at least one member of \{B_1, \ldots, B_n\}”.

So the left side is like and and the right side is like or.

- If $n = 1$, we have the kind of sequent seen before.

- If $n = 0$, it’s as if the right-hand side is $\bot$. 


Sequent Calculus Rules

• Rather than have a sequence of formulas as in ND, all relevant formulas are represented in the sequent.

• A proof progresses from one sequent to another.

• It is easiest to start with a goal sequent and work backward ("upward"), even though the rules are cast as if we were working downward.

• In some cases, more than one sequent is required to prove a lower sequent.
Convention

• In what follows, ____ and … stand for sets of formulas.

• Often these sets are shown as $\Gamma$ and $\Delta$, but our notation seems less cluttered.

• Also ___, F, G means ____ $\cup \{F, G\}$. 
L and R Rules

• For each connective, there is a rule for introducing the connective on the left and one for introducing it on the right.

• These correspond to elimination and introduction rules, respectively, in natural deduction.
SC Rules for $\land$

• Recall the natural deduction (ND) rules $\land E$:

\[
\begin{array}{c}
F \land G \\
\hline
F \\
\end{array}
\quad
\begin{array}{c}
F \land G \\
\hline
G \\
\end{array}
\]

• In SC, the $\land E$ rule corresponds to $\land L$ (L for “left”):

\[
\begin{array}{c}
\_\_, F, G \vdash \ldots \\
\hline
\_\_, F \land G \vdash \ldots \\
\_\_, F, G \land L \vdash \ldots \\
\end{array}
\]
Interpretation of $\land L$

\[
\begin{align*}
\text{___, F, G } & \vdash \ldots \\
\text{___, F } & \land G \vdash \ldots & \land L \\
\text{___, F } & \land G \vdash \ldots
\end{align*}
\]

Meaning: \textbf{If} we can deduce \ldots from \text{___, F, G} \textbf{then} we can also deduce \ldots from \text{___, F } \land G. 

This is indeed what $\land E$ in ND tells us.
Informal: “Information Flow” in an SC Proof

assuming the SC proof is constructed bottom to top
Information Flow in $\land L$

From $F \land G$ we can derive both $F$ and $G$ (in ND).

From $\text{___, F, G}$ we can derive $\ldots$.

From $\text{___, F, G}$ we can derive $\ldots$. 
Rules for $\land R$

• Recall the natural deduction (ND) rule $\land I$:

\[
\begin{array}{c}
F \\
\hline
G
\end{array}
\Rightarrow
F \land G
\]

In SC, the $\land I$ rule corresponds to

\[
___ \vdash F, ... \quad ___ \vdash G, ... \quad (2 \text{ sequents})
\]

\[
\begin{array}{c}
___ \vdash F \land G, ...
\end{array}
\]

$\land R$

Here there two sequents above the line, and each must be proved. The information flow combines $F$ and $G$. Bottom up, one sequent “splits” into two.
∧ Rule Summary

___, A, B |– …

________________________
___ |– A, … ___ |– B, …

“To use A ∧ B, you can use both A and B.”

________________________
___ |– A ∧ B, …

“To derive A ∧ B, derive each of A and B separately.”
SC “Axiom”

• In SC, there is one axiom (rule with no antecedent):

\[
\text{Axiom} \quad \Box, \, \Box, \, P \vdash P, \, \ldots
\]

• Here \(\Box\) and \(\ldots\) represent sets of formulas as usual.

• The meaning in this case is that any formula can be derived from itself.

• A full derivation tree will have an \textit{axiom at each leaf}. 
**v Rule Summary**

\[ \text{___} \vdash A, B, \ldots \]
\[ \text{__________________________ vR} \]
\[ \text{___} \vdash A \lor B, \ldots \]

“To derive \( A \lor B \), it suffices to derive either.”

(like \( \lor \) Introduction in ND)

\[ \text{___, A} \vdash \ldots \quad \text{___, B} \vdash \ldots \]
\[ \text{__________________________ vL} \]
\[ \text{___, A \lor B} \vdash \ldots \]

“To use \( A \lor B \), we need separate derivations using \( A \) and \( B \).”

(similar to \( \lor \) Elmination in ND, but cleaner).
→ Rules

\[ \text{___, A } \vdash \text{ B, ...} \]
\[ \text{__________} \]
\[ \text{___ } \vdash \text{ A } \rightarrow \text{ B, ...} \]

“To derive A → B, it suffices to derive B with A as an added assumption.” (similar to → Introduction in ND).

\[ \text{___ } \vdash \text{ A, ...} \]
\[ \text{___, B } \vdash \text{ ...} \]
\[ \text{__________} \]
\[ \text{___, A } \rightarrow \text{ B } \vdash \text{ ...} \]

“If A can be proved, and we can prove ... based on B as an added assumption, then ... can be proved from A → B.”

(similar to → Elimination in ND).
¬ Rules

___ ,A |- ... 
_______________ ¬R
___ |- ¬A, ...

“To derive ¬A, ... from ___, it suffices to use ___, A to get ...
... .” (similar to ¬ Introduction in ND)

___________________ ¬L

___ ,¬A |- ...

“To use ___, ¬A to derive ... it suffices to derive A, ... from ___.”

(similar to RAA in ND)
A way to remember → Rules

• Recall that F→G is like ¬F ∨ G.
• So the effect of the → can be achieved (bottom up) by using the appropriate ∨ rule followed by a ¬ rule.

• In particular, →L splits as does ∨L. But once the split is done, the F in ¬F can be “flipped” to the other side.
## SC Rule Summary

<table>
<thead>
<tr>
<th></th>
<th>Introduce on Left</th>
<th>Introduce on Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\land$</td>
<td>__, A, B $\vdash \ldots$</td>
<td>__ $\vdash$ A, $\ldots$ __ $\vdash$ B, $\ldots$</td>
</tr>
<tr>
<td></td>
<td>__ __, A $\land$ B $\vdash \ldots$</td>
<td>__ $\vdash$ A $\land$ B, $\ldots$</td>
</tr>
<tr>
<td>$\lor$</td>
<td>__ $\vdash$ A $\ldots$ __ $\vdash$ B $\ldots$ __ __, A $\lor$ B $\vdash \ldots$</td>
<td>__ $\vdash$ A $\lor$ B, $\ldots$ __ __ $\vdash$ A $\lor$ B, $\ldots$</td>
</tr>
<tr>
<td></td>
<td>__ $\vdash$ A $\lor$ B $\ldots$ __ __ __, A $\rightarrow$ B $\vdash \ldots$</td>
<td>__ $\vdash$ A $\rightarrow$ B, $\ldots$ __ __ __ $\vdash$ A $\rightarrow$ B, $\ldots$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>__ $\vdash$ A, $\ldots$ __ __ __ __, $\neg$ A $\vdash \ldots$</td>
<td>__ $\vdash$ A, $\ldots$ __ __ __ __, $\neg$ A, $\ldots$</td>
</tr>
</tbody>
</table>

Assuming “introduce” means from top down, even though we are typically working bottom up.
Other Rules
not dependent on connectives

___ |– …

_____________

___ |– …, A |– …

“Formulas can be added arbitrarily on the left.” (going downward)

___ |– …

_____________

___ |– A, …

“Formulas can be added arbitrarily on the right.” (going downward)
Sequent Calculus Strategy Summary

Working **backward/upward** from the desired sequent:

<table>
<thead>
<tr>
<th>Situation</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A on left and right</td>
<td>Axiom.</td>
</tr>
<tr>
<td>¬A in a right formula</td>
<td>Flip A to the left.</td>
</tr>
<tr>
<td>¬A in a left formula</td>
<td>Flip A to the right.</td>
</tr>
<tr>
<td>A∧B in a right formula</td>
<td><strong>Split</strong> right set into A and B versions.</td>
</tr>
<tr>
<td>A∧B in a left formula</td>
<td>Replace the formula with A, B.</td>
</tr>
<tr>
<td>A∨B in a right formula</td>
<td>Replace the formula with A, B.</td>
</tr>
<tr>
<td>A∨B in a left formula</td>
<td><strong>Split</strong> left set into A and B versions.</td>
</tr>
<tr>
<td>A→B in a right formula</td>
<td>Replace with A on the left, B on the right.</td>
</tr>
<tr>
<td>A→B in a left formula</td>
<td><strong>Split</strong> into two versions with A on the right in one, B on the left in the other.</td>
</tr>
</tbody>
</table>
Example Sequent Calculus Proof

Constructed from bottom to top:

\[ \neg P \vdash Q \]

\[ P \lor Q, \neg P \vdash Q \]
Example Sequent Calculus Proof

Constructed from bottom to top:

\[
\frac{P \lor Q}{P, Q} \\
\frac{P \lor Q}{\neg P} \quad \text{\neg L rule}
\]
Example Sequent Calculus Proof

Constructed from bottom to top:

\[ \begin{align*}
P & \vdash P, Q \\
Q, & \vdash P, Q \\
P \lor Q & \vdash P, Q \\
P \lor Q, \neg P & \vdash \neg Q
\end{align*} \]
Example Sequent Calculus Proof

Constructed from bottom to top:

\[ \begin{array}{c}
\vdash P, Q \\
\vdash Q, \neg P, Q \\
\vdash P, Q, \neg P \vdash \bot \\
\vdash P \vee Q \vdash \bot \\
\vdash P \vee Q, \neg P \vdash \bot \\
\end{array} \]

\[ \begin{array}{c}
P \vee Q \\
\neg L \text{ rule} \\
Q \\
vL \text{ rule} \\
P \vee Q,\neg P \vdash Q \\
\neg L \text{ rule} \\
P \vee Q,\neg P \vdash \bot \\
\neg E \\
P \vdash \bot \\
\bot E \\
\neg P \\
\bot E \\
\vdash P \\
vE_{1,2} \\
\end{array} \]

Corresponding Natural Deduction Tree:
Rough Correspondence: SC vs. ND

- The last sequent of an SC derivation will always correspond to the overall sequent derived in ND.

- Other sequents may correspond to subproofs.

- Moving from bottom to top in an SC proof is like working outside-in in a ND proof.

- Consider LHS of a sequent to be all operative hypotheses, including assumptions in sub-proofs.

- Consider RHS to be goal (as a disjunction).
Conversion Between SC and ND

Compare JAPE SC Version (MCS)

\[
\begin{align*}
\text{Axiom} & \quad P \vdash P, Q & \quad Q, \vdash P, Q \\
\text{v \vdash rule} & \quad P \lor Q \vdash P, Q \\
\text{\neg \vdash rule} & \quad P \lor Q, \neg P \vdash \neg Q
\end{align*}
\]

MCS = Multi-Conclusion (classical)
SCS = Single Conclusion (intuitionistic)
More Sequent Calculus Examples
(constructed working backward/upward)

\[ \vdash ((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R) \]
More Sequent Calculus Examples

\[
(P \rightarrow Q) \land (Q \rightarrow R) \vdash (P \rightarrow R) \quad \vdash \rightarrow \\
\vdash ((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R)
\]
More Sequent Calculus Examples

\[(P \rightarrow Q), (Q \rightarrow R) \vdash (P \rightarrow R)\]
\[(P \rightarrow Q) \land (Q \rightarrow R) \vdash (P \rightarrow R)\]
\[\vdash ((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R)\]
More Sequent Calculus Examples

\[(P \rightarrow Q), (Q \rightarrow R), P \vdash R\]
\[(P \rightarrow Q), (Q \rightarrow R) \vdash (P \rightarrow R)\]
\[(P \rightarrow Q) \land (Q \rightarrow R) \vdash (P \rightarrow R)\]
\[\vdash ((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R)\]
More Sequent Calculus Examples

\[(P \rightarrow Q), R, P \vdash R \quad (P \rightarrow Q), P \vdash Q, R \rightarrow \vdash\]
\[(P \rightarrow Q), (Q \rightarrow R), P \vdash R \quad \vdash \rightarrow\]
\[(P \rightarrow Q), (Q \rightarrow R) \vdash (P \rightarrow R) \quad \wedge \vdash\]
\[(P \rightarrow Q) \wedge (Q \rightarrow R) \vdash (P \rightarrow R) \quad \vdash \rightarrow\]
\[\vdash ((P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow (P \rightarrow R)\]
More Sequent Calculus Examples

\[
\begin{align*}
\text{Ax} & \quad P \vdash Q, R, P \\
(P \rightarrow Q), R, P & \vdash R \\
&P \rightarrow Q, R, P \\
(P \rightarrow Q), (Q \rightarrow R), P & \vdash R \\
&P \rightarrow Q, (Q \rightarrow R) \\
(P \rightarrow Q) & \wedge (Q \rightarrow R) \\
\vdash (P \rightarrow Q) \wedge (Q \rightarrow R) \\
\vdash (P \rightarrow Q, R) & \rightarrow (P \rightarrow R) \\
\end{align*}
\]
More Sequent Calculus Examples

\[
\begin{align*}
\text{Ax} & \quad \text{Ax} \\
\text{Ax} & \quad \text{Ax} \\
\text{Ax} & \quad \text{Ax} \\
\text{Ax} & \quad \text{Ax} \\
\text{Ax} & \quad \text{Ax} \\
\end{align*}
\]

\[
\begin{align*}
&(P \rightarrow Q), R, P \vdash R & (P \rightarrow Q), P \vdash Q, R & \rightarrow \vdash \\
&\vdash (P \rightarrow Q), (Q \rightarrow R), P \vdash R & \vdash (P \rightarrow Q), (Q \rightarrow R) \vdash (P \rightarrow R) & \vdash \rightarrow \\
&\vdash (P \rightarrow Q) \land (Q \rightarrow R) \vdash (P \rightarrow R) & \vdash \rightarrow \\
&\vdash ((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R)
\end{align*}
\]
An SC Prover in Prolog

% If L |- R, where L and R are lists of formula,
% prove(L, R, Proof) will produce a sequent calculus proof tree.
% Using test(L, R) will display the proof in line-numbered fashion.
% If the sequent is not provable, the prove predicate will fail.

prove(L, R, axiom(L, R)) :- member(F, L), member(F, R).

prove(L, R, notLeft(L, R, Proof)) :-
   select(not(A), L, Rest),
   prove(Rest, [A | R], Proof).

prove(L, R, notRight(L, R, Proof)) :-
   select(not(A), R, Rest),
   prove([A | L], Rest, Proof).

prove(L, R, andLeft(L, R, Proof)) :-
   select(and(A, B), L, Rest),
   prove([A, B | Rest], Rest, Proof).

prove(L, R, orRight(L, R, Proof)) :-
   select(or(A, B), R, Rest),
   prove(L, [A, B | Rest], Rest, Proof).

prove(L, R, impliesRight(L, R, Proof)) :-
   select(implies(A, B), R, Rest),
   prove([A | L], [B | Rest], Proof).

prove(L, R, andRight(L, R, ProofL, ProofR)) :-
   select(and(A, B), R, Rest),
   prove(L, [A | Rest], ProofL),
   prove(L, [B | Rest], ProofR).

prove(L, R, orLeft(L, R, ProofL, ProofR)) :-
   select(or(A, B), L, Rest),
   prove([A | Rest], R, ProofL),
   prove([B | Rest], R, ProofR).

prove(L, R, impliesLeft(L, R, ProofL, ProofR)) :-
   select(implies(A, B), L, Rest),
   prove(Rest, [A | R], L, ProofL),
   prove([B | Rest], R, ProofR).
/* Example output from test([not(or(a, b))], [and(not(a), not(b))]).
   
   Proof of sequent or(not(a),not(b)) |- not(and(a,b)):
   1: a, b |- a                           -- Axiom
   2: not(a), a, b |-                     -- not L, line 1
   3: a, b |- b                           -- Axiom
   4: not(b), a, b |-                     -- not L, line 3
   5: a, b, or(not(a),not(b)) |-          -- or L, lines 2 and 4
   6: and(a,b), or(not(a),not(b)) |-      -- and L, line 5
   7: or(not(a),not(b)) |- not(and(a,b))  -- not R, line 6
*/

We use line numbers to display the tree linearly.
JAPE SCS vs. MCS

- SCS: Single-conclusion:
  Will only prove intuitionistic sequents.
  The RHS is always a single formula.
  For some reason
  \(-\varphi\) converts to \(\varphi \rightarrow \bot\) first.

This is \(-\neg\) Introduction.
\(-\neg\) Elimination cannot be proved intuitionistically.
SC vs. Tableaux

- Tableaux proofs were introduced previously.
- A correspondence can be established between an SC proof and a **block tableau** proof $T$.
- Tableau is constructed top-down; SC is bottom-up.
- The conclusion of SC is negated in $T$.
- The premises of SC, if any, are unnegated in $T$.
- There is a correspondence between rules of the two systems.
- Splitting in SC is like splitting in the block tableau.
- Axioms of SC correspond to path closure.
### Sequent Calculus vs. Tableaux

<table>
<thead>
<tr>
<th>Sequent Calculus</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructed bottom-up.</td>
<td>Constructed top-down.</td>
</tr>
<tr>
<td>Proves $A_1, \ldots, A_m \vdash B_1, \ldots, B_n$</td>
<td>Negated conclusion (typically $m = 0, n = 1$)</td>
</tr>
<tr>
<td>Equivalent to proving $(\neg A_1 \lor \ldots \lor \neg A_m \lor B_1 \lor \ldots \lor B_n)$</td>
<td>$\neg (\neg A_1 \lor \ldots \lor \neg A_m \lor B_1 \lor \ldots \lor B_n)$ $\equiv A_1 \land \ldots \land A_m \land \neg B_1 \land \ldots \land \neg B_n$</td>
</tr>
<tr>
<td>Formulas on the right</td>
<td>Originally negated formulas</td>
</tr>
<tr>
<td>Formulas on the left</td>
<td>Originally un-negated formulas</td>
</tr>
<tr>
<td>Axiom ..., $p$, ...</td>
<td>- __, $p$, __</td>
</tr>
</tbody>
</table>
Tableau vs. SC Example

Negated formulas in tableau go on the right of $\vdash$.

Unnegated formulas in tableau go on the right of $\vdash$.

<table>
<thead>
<tr>
<th>Block Tableau</th>
<th>SC “upside-down”</th>
</tr>
</thead>
<tbody>
<tr>
<td>${ \neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) }$</td>
<td>$\vdash (p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$</td>
</tr>
</tbody>
</table>
### Tableau vs. SC Example

<table>
<thead>
<tr>
<th>Block Tableau</th>
<th>SC “upside-down”</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ \neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) }</td>
<td>\vdash (p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg (\neg q \rightarrow \neg p) }</td>
<td>(p \rightarrow q) \vdash (\neg q \rightarrow \neg p)</td>
</tr>
</tbody>
</table>
# Tableau vs. SC Example

<table>
<thead>
<tr>
<th>Block Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ \neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) }</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg(\neg q \rightarrow \neg p) }</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg q, \neg \neg p }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SC “upside-down”</th>
</tr>
</thead>
<tbody>
<tr>
<td>\vdash (p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)</td>
</tr>
<tr>
<td>(p \rightarrow q) \vdash (\neg q \rightarrow \neg p)</td>
</tr>
<tr>
<td>\neg q, (p \rightarrow q) \vdash \neg p</td>
</tr>
</tbody>
</table>
## Tableau vs. SC Example

<table>
<thead>
<tr>
<th>Block Tableau</th>
<th>SC “upside-down”</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ \neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) }</td>
<td>\neg (p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg (\neg q \rightarrow \neg p) }</td>
<td>(p \rightarrow q) \neg (\neg q \rightarrow \neg p)</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg q, \neg \neg p }</td>
<td>\neg q, (p \rightarrow q) \neg \neg p</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg q, p }</td>
<td>\neg q, (p \rightarrow q), p \neg</td>
</tr>
</tbody>
</table>

# Tableau vs. SC Example

<table>
<thead>
<tr>
<th>Block Tableau</th>
<th>SC “upside-down”</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ \neg ((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) }</td>
<td>\neg (p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg(\neg q \rightarrow \neg p) }</td>
<td>(p \rightarrow q) \neg (\neg q \rightarrow \neg p)</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg q, \neg \neg p }</td>
<td>\neg q, (p \rightarrow q) \neg q</td>
</tr>
<tr>
<td>{ (p \rightarrow q), \neg q, p }</td>
<td>\neg q, (p \rightarrow q), p</td>
</tr>
<tr>
<td>{ \neg p, \neg q, p }X</td>
<td>\neg q, q, p</td>
</tr>
<tr>
<td>{ q, \neg q, p }X</td>
<td>q, p</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

\[ p \rightarrow q \]

\[ \neg q \]

\[ \neg p \]
One Sequent: \( \neg(\neg E \land \neg F) \vdash E \lor F \)

Proved Five Ways

Natural Deduction (tree):

\[
\begin{align*}
\neg(\neg(E \lor F)) & \quad \text{[E]}^2 \\
(E \lor F) & \quad \text{[F]}^3
\end{align*}
\]

\[
\begin{aligned}
\bot & \quad \neg I_2 \\
\neg E & \quad \bot \\
\neg F & \quad \neg I_3 \\
\neg E \land \neg F & \quad \land I \\
\neg E & \quad \neg E \\
\bot & \quad \text{RAA}_1
\end{aligned}
\]

\[
\begin{aligned}
\bot & \quad \text{EvF}
\end{aligned}
\]
One Sequent  \( \neg(\neg E \land \neg F) \vdash E \lor F \)

Proved Five Ways

Natural Deduction (Fitch diagram)
One Sequent
\[ \neg(\neg E \land \neg F) \vdash E \lor F \]

Proved Five Ways

Sequent Calculus (read bottom-up)

\[ 
\begin{array}{ll}
E & \vdash E, F \\
F & \vdash E, F \\
\hline & \vdash E, F, \neg E \\
\hline & \vdash E, F, \neg F \\
\hline & \vdash E, F, (\neg E \land \neg F) \\
\hline & \vdash (\neg E \land \neg F) \\
\hline & \vdash E, F \\
\hline & \vdash (\neg E \land \neg F) \\
\vdash E \lor F \\
\end{array} 
\]

Here the correspondence with the Natural Deduction proof is not 1-1.
One Sequent  \( \neg(\neg E \land \neg F) \vdash E \lor F \)

Proved Five Ways

Block Tableau

\[
\begin{align*}
\{ \neg(\neg E \land \neg F), \neg(E \lor F) \} \\
\{ \neg(\neg E \land \neg F), \neg E, \neg F \} \\
\{ \neg E \lor \neg F, \neg E, \neg F \} \\
\{ \neg E, \neg E, \neg F \} & \quad \{ \neg F, \neg E, \neg F \} \\
\{ E, \neg E, \neg F \} & \quad \{ F, \neg E, \neg F \} \\
X & \quad X
\end{align*}
\]

The blocks could close one step earlier, due to \( \neg \neg E \) and \( \neg E \), etc.
One Sequent \( \neg(\neg E \land \neg F) \vdash E \lor F \)

Proved Five Ways

Tableau (tree form)

1. \( \neg(\neg E \land \neg F) \) premise \( \checkmark \)
2. \( \neg(E \lor F) \) negated conclusion \( \checkmark \)
3. \( \neg E \) (2 stack)
4. \( \neg F \) (2 stack)
5. \( \neg
\neg E \) (1 split) \( \checkmark \)
6. \( \neg
\neg F \) (1 split) \( \checkmark \)
7. \( E \) (5 \( \neg \neg \))
   \[ X \, (3, 7) \]
8. \( F \) (5 \( \neg \neg \))
   \[ X \, (4, 8) \]

The paths could close one step earlier, due to \( \neg
\neg E \) and \( \neg E \), etc.

Please use for *checking*, not doing, homework!

**Sequent Prover (seqprover)**

Sequent: \((p \rightarrow q) / (q \rightarrow r) \rightarrow p \rightarrow r\)

Output style: **Pretty form**

[Top page]

Trying to prove with threshold = 0

\[
\begin{align*}
\text{Ax} & \quad \text{Ax} & \quad \text{Ax} \\
p \rightarrow q, p, r & \quad p, r \rightarrow p, r & \quad p, q \rightarrow q, r & \quad p, q, r \rightarrow r \\
\hline
\end{align*}
\]

\[
\begin{align*}
\text{L} & \quad \text{L} & \quad \text{L} \\
p, q \rightarrow r \rightarrow p, r & \quad p, q, q \rightarrow r \rightarrow r & \quad \hline
\end{align*}
\]

\[
\begin{align*}
\text{R} & \quad \hline
p \rightarrow q, q \rightarrow r \rightarrow r & \quad p \rightarrow r \\
\hline
\end{align*}
\]

\[
\begin{align*}
\text{L} \wedge & \quad \hline
(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow p \rightarrow r \\
\end{align*}
\]

# Proved in 0 msec.
Tableau Prover
http://www.umsu.de/logik/trees/
Please use for *checking*, not doing, homework!

Tree Proof Generator

```
((p \to q) \to (\neg q \to \neg p))
```

```
((p\to q)\to(\neg q\to \neg p)) is valid.
```

1. \(\neg((p\to q)\to(\neg q\to \neg p))\)
2. \((p\to q)\) (1)
3. \(\neg(\neg q\to \neg p)\) (1)
4. \(\neg q\) (3)
5. \(\neg p\) (3)
6. \(\neg p\) (2)
7. \(q\) (2)