Sequent Calculus

Robert Keller

January 2011

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. This is similar to the **contextual representation** used in the soundness proof. However, some authors use ⇒ or → instead of |–.
- It is easier to **implement in software** a proof generator based on SC than it is one based on ND.

Meaning of |— in SC

- The *intuitive* meaning of

  $A_1, \ldots, A_m$ |— $B_1, \ldots, B_n$

  "Using hypotheses in the set $\{A_1, \ldots, A_m\}$ one can derive **at least one** member of $\{B_1, \ldots, B_n\}$".
- If n = 1, we have the kind of sequent we've seen before.
- If n = 0, it's like the right-hand side is ⊥.

Sequent Calculus Rules

- Rather than have a sequence of formulas as in ND, all relevant formulas are represented in the sequent, as in the **contextual form**.
- A proof progresses from one sequent to another.
- It is easiest to start with a goal sequent and work backward ("upward").
- 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 Γ and Δ, but our notation seems less cluttered.
- Also ___, F, G means ___ ∪ {F, G}.

Rules for ∧

- Recall the natural deduction (ND) rules ∧E:

$$\frac{F \wedge G}{F} \qquad \frac{F \wedge G}{G}$$

- In SC, the ∧E rule corresponds to ∧L (L for "left"):

$$\frac{\_\_\_,\ F,\ G \ |\!\!-\ \ldots}{\_\_\_,\ F \wedge G \ |\!\!-\ \ldots} \quad \wedge L$$

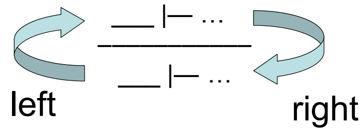
Meaning of ∧L

___, F, G |— …
―――――――――――
___, F ∧ G |— … ∧L

Meaning: If we can deduce … from ___, F, G

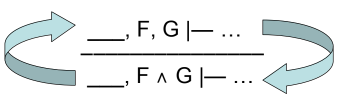
then we can also deduce … from ___, F ∧ G.

“Information Flow” in an SC Proof



assuming the SC proof is constructed bottom to top

Information Flow in ∧L



∧L

From F ∧ G we can derive both F and G (in ND).

From ___, F, G we can derive … .

Rules for ∧R

- Recall the natural deduction (ND) rule ∧I:

  F G
  ――――
  F ∧ G

  In SC, the ∧I rule corresponds to

  ___ |— F, … ___ |— G, … (2 sequents)
  ――――――――――――――――
  ___ |— F ∧ G, … ∧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 |– … ∧L

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

___ |– A, … ___ |– B, …
――――――――――――――
___ |– A ∧ B, … ∧R

“To ***derive*** A ∧ B, derive each separately.”

SC “Axiom”

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

  ―――――――― Axiom
  ___, P |– P, …

- Here ___ and … 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 **axiom at each leaf**.

∨ Rule Summary

___ |– A, B, …
——————— ∨R
___ |– A ∨ B, …

"To ***derive*** A ∨ B, it suffices to derive either."
(like ∨ Introduction in ND)

___, A |– …   ___, B |– …
——————————————— ∨L
___, A ∨ B |– …

"To *use* A ∨ B, we need separate derivations **using** A and B."
(similar to ∨ Elmination in ND).

→ Rules

___, A |– B, …
——————————— →R
___ |– A → B, …

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

___ |– A, …   ___, B |– …
——————————————— →L
___, A → B |– …

"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)

___ |– A, …
——————— ¬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

	Introduce on Left	Introduce on Right
∧	___, A, B |– … ———————— ___, A ∧ B |– …	___ |– A, … ___ |– B, … ———————————— ___ |– A ∧ B, …
∨	___, A |– … ___, B |– … ———————————— ___, A ∨ B |– …	___ |– A, B, … ———————— ___ |– A ∨ B, …
→	___ |– A, … ___, B |– … ———————————— ___, A → B |– …	___, A |– B, … ———————— ___ |– A → B, …
¬	___ |– A, … ———————— ___ ,¬A |– …	___ ,A |– … ———————— ___ |– ¬A, …

Assuming "introduce" means from top down,
even though we are typically working bottom up.

Sequent Calculus Strategy Summary

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

Situation	Action
A on left and right	Axiom.
¬A in a right formula	Flip A to the left.
¬A in a left formula	Flip A to the right.
A∧B in a right formula	**Split** right set into A and B versions.
A∧B in a left formula	Replace the formula with A, B.
A∨B in a right formula	Replace the formula with A, B.
A∨B in a left formula	**Split** left set into A and B versions.
A→B in a right formula	Replace with A on the left, B on the right.
A→B in a left formula	**Split** into two versions with A on the right in one, B on the left in the other.

Example Sequent Calculus Proof

Constructed from bottom to top:

P v Q, ¬P |– Q

Example Sequent Calculus Proof

Constructed from bottom to top:

P v Q |– P, Q ¬L rule

P v Q, ¬P |–Q

Example Sequent Calculus Proof

Constructed from bottom to top:

P |– P, Q Q, |– P, Q vL rule

P v Q |– P, Q ¬L rule

P v Q, ¬P |–Q

Example Sequent Calculus Proof

Constructed from bottom to top:

________ Axiom ________ Axiom

P |– P, Q Q, |– P, Q vL rule

P v Q |– P, Q ¬L rule

P v Q, ¬P |–Q

Corresponding Natural Deduction Tree:

$[P]_1$ ¬P

⊥ ¬E

P v Q Q $[Q]_2$ $vE_{1,2}$

Q

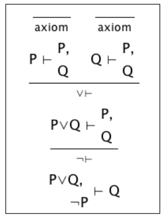
Rough Correspondence: SC vs. ND

- 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

- http://twelf.plparty.org/wiki/POPL_Tutorial/Sequent_vs_Natural_Deduction
- http://www.ags.uni-sb.de/~chris/papers/2002-pisa.pdf

Compare JAPE SC Version (MCS)



________ Axiom   ________ Axiom
P |–P, Q   Q, |– P, Q   v |– rule
P v Q |– P, Q   ¬ |– rule
P v Q, ¬P |–Q

MCS = Multi-Conclusion (classical)
SCS = Single Conclusion (intuitionistic)

More Sequent Calculus Examples
(constructed working backward/upward)

|– ((P → Q) ∧ (Q → R)) → (P → R)

More Sequent Calculus Examples

(P → Q) ∧ (Q → R) |– (P → R) |- →
|– ((P → Q) ∧ (Q → R)) → (P → R)

More Sequent Calculus Examples

(P → Q), (Q → R) |– (P → R)   ∧ |-
(P → Q) ∧ (Q → R) |– (P → R) |- →
|– ((P → Q) ∧ (Q → R)) → (P → R)

More Sequent Calculus Examples

(P → Q), (Q → R), P |– R |- →
(P → Q), (Q → R) |– (P → R)   ∧ |-
(P → Q) ∧ (Q → R) |– (P → R) |- →
|– ((P → Q) ∧ (Q → R)) → (P → R)

More Sequent Calculus Examples

(P → Q), R, P |– R   (P → Q), P |– Q, R   → |-
(P → Q), (Q → R), P |– R |- →
(P → Q), (Q → R) |– (P → R)   ∧ |-
(P → Q) ∧ (Q → R) |– (P → R) |- →
|– ((P → Q) ∧ (Q → R)) → (P → R)

More Sequent Calculus Examples

```
______________ Ax   P |– Q, R, P         Q, P |– Q, R     → |-
(P → Q), R, P |– R            (P → Q), P |– Q, R           → |-
(P → Q), (Q → R), P |– R                   |- →
(P → Q), (Q → R) |– (P → R)                ∧ |-
(P → Q) ∧ (Q → R) |– (P → R)               |- →
|– ((P → Q) ∧ (Q → R)) → (P → R)
```

More Sequent Calculus Examples

```
                                ________ Ax    _________ Ax
______________ Ax   P |– Q, R, P         Q, P |– Q, R     → |-
(P → Q), R, P |– R            (P → Q), P |– Q, R           → |-
(P → Q), (Q → R), P |– R                   |- →
(P → Q), (Q → R) |– (P → R)                ∧ |-
(P → Q) ∧ (Q → R) |– (P → R)               |- →
|– ((P → Q) ∧ (Q → R)) → (P → R)
```

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], R, Proof).

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

Formatted Output of the SC Prover

```
% Propositional Sequent Calculus Prover constructed in Prolog
% author: Robert Keller

/* 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

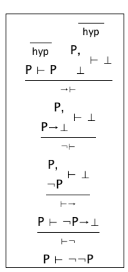
- SCS: Single-conclusion:

  Will only prove intuitionistic sequents.

  The RHS is always a single formula.

  For some reason

  ¬φ converts to φ→⊥ first.



This is ¬¬ Introduction.

¬¬ Elimination cannot be proved intuitionistically.

SC vs. Tableaux

- Tableaux proofs are introduced elsewhere. Skip this for now if they haven't been covered.
- 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

Sequent Calculus	Tableau
Constructed bottom-up.	Constructed top-down.
Proves $A_1, \ldots, A_m \vdash B_1, \ldots, B_n$ Equivalent to proving $(\neg A_1 \vee \ldots \vee \neg A_m \vee B_1 \vee \ldots \vee B_n)$	Negated conclusion (typically m = 0, n = 1) $\neg(\neg A_1 \vee \ldots \vee \neg A_m \vee B_1 \vee \ldots \vee B_n)$ $\equiv A_1 \wedge \ldots \wedge A_m \wedge \neg B_1 \wedge \ldots \wedge \neg B_n$
Formulas on the right	Originally negated formulas
Formulas on the left	Originally un-negated formulas
Axiom …, p, … |- __, p, __	Closure p, ¬p

Tableau vs. SC Example

negated formulas in tableau go on the right of |–
unnegated formulas in tableau go on the right of |–

Block Tableau	SC "upside-down"
{ ¬((p → q) → (¬q → ¬p)) }	|– (p → q) → (¬q → ¬p)

Tableau vs. SC Example

Block Tableau	SC "upside-down"
{ ¬((p → q) → (¬q → ¬p)) }	|– (p → q) → (¬q → ¬p)
{ (p → q), ¬(¬q → ¬p) }	(p → q) |– (¬q → ¬p)

Tableau vs. SC Example

Block Tableau	SC "upside-down"
{ ¬((p → q) → (¬q → ¬p)) }	|– (p → q) → (¬q → ¬p)
{ (p → q), ¬(¬q → ¬p) }	(p → q) |– (¬q → ¬p)
{ (p → q), ¬q, ¬¬p }	¬q, (p → q) |– ¬p

Tableau vs. SC Example

Block Tableau	SC "upside-down"
{ ¬((p → q) → (¬q → ¬p)) }	|– (p → q) → (¬q → ¬p)
{ (p → q), ¬(¬q → ¬p) }	(p → q) |– (¬q → ¬p)
{ (p → q), ¬q, ¬¬p }	¬q, (p → q) |– ¬p
{ (p → q), ¬q, p }	¬q, (p → q), p |–

Tableau vs. SC Example

Block Tableau	SC "upside-down"
{ ¬((p → q) → (¬q → ¬p)) }	|– (p → q) → (¬q → ¬p)
{ (p → q), ¬(¬q → ¬p) }	(p → q) |– (¬q → ¬p)
{ (p → q), ¬q, ¬¬p }	¬q, (p → q) |– ¬p
{ (p → q), ¬q, p }	¬q, (p → q), p |–
{¬p, ¬q, p }X { q, ¬q, p }X	¬q, q, p |– q, p |– p(ax)
	q, p |– q (ax)

What are Blocks?

- A block is simply a set of formulas corresponding to the ***un-checked*** formulas on a **single path** in the tree tableau.

- We start with one block.
- **Stacking** replaces formulas **within** that block.
- **Splitting** splits the block into two.
- **Closure** is when a block contains a formula and its negation.

Block Tableau from Tableau Example
Negated Formula at Root

1. $\neg((p \to q) \to (\neg q \to \neg p))$

Corresponding Block Tableau

$\{ \neg((p \to q) \to (\neg q \to \neg p)) \}$

---

1. $\neg((p \to q) \to (\neg q \to \neg p))$ ✓
2. $(p \to q)$
3. $\neg(\neg q \to \neg p)$

Corresponding Block Tableau

$\{ (p \to q), \neg(\neg q \to \neg p) \}$

---

1. $\neg((p \to q) \to (\neg q \to \neg p))$ ✓
2. $(p \to q)$
3. $\neg(\neg q \to \neg p)$ ✓
4. $\neg q$
5. $\neg\neg p$

Corresponding Block Tableau

$\{ (p \to q), \neg q, \neg\neg p \}$

---

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

Corresponding Block Tableau

$\{(p \to q), \neg q, p \}$

---

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

Split from line 2.

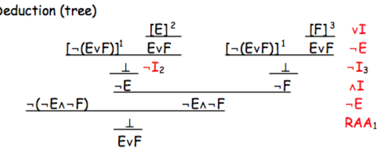
7. $\neg p$ X(6, 7)
8. q X(4, 8)

Corresponding Block Tableau Splits

$\{\neg p\ \neg q, p \}$, $\{q, \neg q, p \}$ both blocks close

One Sequent ¬(¬E∧¬F) |- E∨F
Proved Five Ways

Natural Deduction (tree)



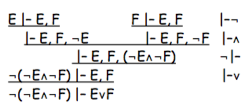
One Sequent ¬(¬E∧¬F) |- E∨F
Proved Five Ways

Natural Deduction (Fitch diagram)

1	¬(¬E∧¬F)	premise
2	¬(E∨F)	assumption
3	E	assumption
4	E∨F	3, ∨I
5	⊥	2, 4, ¬E
6	¬E	3-5, ¬I
7	F	assumption
8	E∨F	7, ∨I
9	⊥	2, 8, ¬E
10	¬F	7-9, ¬I
11	¬E∧¬F	6, 10, ∧I
12	⊥	1, 11, ¬E
13	E∨F	2-12, RAA

One Sequent ¬(¬E∧¬F) |- E∨F
Proved Five Ways

Sequent Calculus (read bottom-up)



Here the correspondence with the Natural Deduction proof is not 1-1.

One Sequent ¬(¬E∧¬F) |- E∨F
Proved Five Ways

Block Tableau

{ ¬(¬E∧¬F), ¬(E∨F) }
{ ¬(¬E∧¬F), ¬E, ¬F }
{ ¬¬E∨¬¬F, ¬E, ¬F }
{ ¬¬E, ¬E, ¬F }   { ¬¬F, ¬E, ¬F }
{ E, ¬E, ¬F }   {F, ¬E, ¬F }
X   X

The blocks could close one step earlier, due to ¬¬E and ¬E, etc.

One Sequent ¬(¬E∧¬F) |- E∨F
Proved Five Ways

Tableau (tree form)

1. ¬(¬E∧¬F)   premise √
2. ¬(E∨F)   negated conclusion √
3. ¬E   (2 stack)
4. ¬F   (2 stack)
5. ¬¬E (1 split) √   6. ¬¬F (1 split) √
7. E (5 ¬¬)   8. F (5¬¬)
X (3, 7)   X (4, 8)

The paths could close one step earlier, due to ¬¬E and ¬E, etc.

An Automated Sequent Calculus Prover:
Defunct? http://bach.istc.kobe-u.ac.jp/seqprover/

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

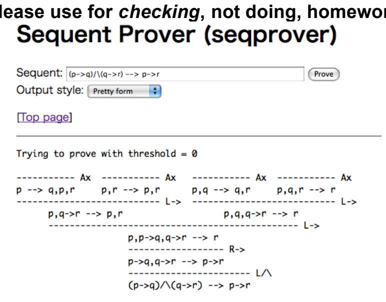


Tableau Prover

http://www.umsu.de/logik/trees/

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