Sequent Calculus & PVS

Outline

- Review
- Order of precedence & logical operators in PVS
- Sequent Calculus
- PVS commands: (FLATTEN), (SPLIT) & (BDDSIMP)
- Checking validity of arguments
- Checking consistency of premises
- Unprovable sequents & counter examples

Review: Key Results used by PVS

Commutative & Associative rules for $\wedge, \vee$

Implication: $\models (\phi \rightarrow \psi) \leftrightarrow \neg\phi \vee \psi$

Iff: $\models (\phi \leftrightarrow \psi) \leftrightarrow (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$

Double negation: $\models \phi \leftrightarrow \neg(\neg\phi)$

Identity rules: $\models \phi \wedge \top \leftrightarrow \phi$, $\models \phi \vee \bot \leftrightarrow \phi$

Dominance rules: $\models \phi \vee \top \leftrightarrow \top$, $\models \phi \wedge \bot \leftrightarrow \bot$

Rule of adjunction: $\wedge i$
$\Gamma \vdash \psi \wedge \chi$ iff $\Gamma \vdash \psi$ and $\Gamma \vdash \chi$

Rule of alternative proof: $\vee e$
$\Gamma, \phi \vee \psi \vdash \chi$ iff $\Gamma, \phi \vdash \chi$ and $\Gamma, \psi \vdash \chi$

and Theorems:

Deduction Theorem: $\Gamma, \phi \vdash \psi$ iff $\Gamma \vdash \phi \rightarrow \psi$

Completeness & Consistency: $\Gamma \vdash \psi$ iff $\Gamma \models \psi$

	introduction	elimination
$\wedge$	$\dfrac{\phi \quad \psi}{\phi \wedge \psi} \wedge i$	$\dfrac{\phi \wedge \psi}{\phi} \wedge e_1$ $\quad$ $\dfrac{\phi \wedge \psi}{\psi} \wedge e_2$
$\vee$	$\dfrac{\phi}{\phi \vee \psi} \vee i_1$ $\quad$ $\dfrac{\psi}{\phi \vee \psi} \vee i_2$	$\dfrac{\phi \vee \psi \quad \boxed{\begin{matrix}\phi \\ \vdots \\ \chi\end{matrix}} \quad \boxed{\begin{matrix}\psi \\ \vdots \\ \chi\end{matrix}}}{\chi} \vee e$

	introduction	elimination
$\rightarrow$	$\dfrac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \psi\end{array}}}{\phi \rightarrow \psi} \rightarrow i$	$\dfrac{\phi \quad \phi \rightarrow \psi}{\psi} \rightarrow e$
$\leftrightarrow$	$\dfrac{\phi \rightarrow \psi \quad \psi \rightarrow \phi}{\phi \leftrightarrow \psi} \leftrightarrow i$	$\dfrac{\phi \leftrightarrow \psi}{\phi \rightarrow \psi} \leftrightarrow e_1 \quad \dfrac{\phi \leftrightarrow \psi}{\psi \rightarrow \phi} \leftrightarrow e_2$

	introduction	elimination
$\neg$	$\dfrac{\boxed{\begin{array}{c}\phi \\ \vdots \\ \bot\end{array}}}{\neg\phi}\ \neg i$	$\dfrac{\phi \quad \neg\phi}{\bot}\ \neg e$
$\neg\neg$	$\dfrac{\phi}{\neg\neg\phi}\ \neg\neg i$	$\dfrac{\neg\neg\phi}{\phi}\ \neg\neg e$
$\bot$	see $\neg e$	$\dfrac{\bot}{\phi}\ \bot e$

Additional Proof Rules

$$\frac{\phi \rightarrow \psi}{\neg\phi \vee \psi} \rightarrow 2\vee \qquad \frac{\neg\phi \vee \psi}{\phi \rightarrow \psi} \vee 2 \rightarrow$$

$$\frac{\phi \rightarrow \psi \qquad \neg\psi}{\neg\phi}\text{MT}$$

$$\frac{\boxed{\begin{array}{c}\neg\phi \\ \vdots \\ \bot\end{array}}}{\phi}\ \text{RAA} \qquad \frac{}{\phi \vee \neg\phi}\text{LEM}$$

Order of Precedence in PVS

Recall: We use precedence of logical connectives and associativity of $\wedge, \vee, \leftrightarrow$ to drop parentheses it is understood that this is shorthand for the fully parenthesized expressions.

Rubin uses order of precedence:

$$\neg, \begin{matrix} \wedge \\ \vee \end{matrix}, \begin{matrix} \rightarrow \\ \leftrightarrow \end{matrix}$$

PVS uses order of precedence:

$$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$$

Logical Operators in PVS

Propositional constants and variables have type “bool” in PVS

bool={TRUE, FALSE}

$\neg$ - NOT, not

$\wedge$ - AND, and, &

$\vee$ - OR, or

$\rightarrow$ - IMPLIES, implies, =>

$\leftrightarrow$ - IFF, iff, <=>

Sequent Calculus

$\phi_1, \phi_2, \ldots, \phi_n \vdash \psi_1 \vee \psi_2 \vee \ldots \vee \psi_m$
is another way of stating
$\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n \vdash \psi_1 \vee \psi_2 \vee \ldots \vee \psi_m$

In sequent calculus it is written as:

$$\begin{array}{|c} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \\ \hline \psi_1 \\ \psi_2 \\ \vdots \\ \psi_m \end{array}$$

There are implicit $\wedge$'s between the premises and implicit $\vee$'s between the conclusions.

Assuming all the ϕ_i's are true, we are trying to prove at least one ψ_j is true.

Def: We call $\phi_1 \wedge \ldots \phi_n \rightarrow \psi_1 \vee \ldots \psi_m$ the *characteristic formula* for the sequent because it is a tautology iff $\phi_1, \ldots, \phi_n \vdash \psi_1 \vee \ldots \psi_m$

Proofs in Sequent Calculus

Proofs are done by transforming the sequent until one of the following forms is obtained:

$$
\begin{array}{|l}
\ldots \\
\phi \\
\hline
\phi \\
\ldots
\end{array}
\quad \text{i.e. } \Gamma, \phi \vdash \phi \vee \ldots
$$

which is a case of Rule Premise and $\vee i_1$

$$\begin{array}{|l} \ldots \\ \hline \top \\ \ldots \end{array} \quad \text{i.e. } \Gamma \vdash \top \vee \ldots$$

which is a case of Dominance of $\top$

$$\begin{array}{|l} \ldots \\ \bot \\ \hline \ldots \end{array} \quad \text{i.e. } \Gamma, \bot \vdash \ldots$$

Which is a case of $\bot e$.

Sequent Calculus Special Cases

No premises: $\begin{array}{|l} \hline \psi_1 \\ \vdots \\ \psi_m \end{array}$ iff $\vdash \psi_1 \vee \ldots \vee \psi_m$

No conclusions: $\begin{array}{|l} \phi_1 \\ \vdots \\ \phi_n \\ \hline \end{array}$ iff $\phi_1 \wedge \ldots \wedge \phi_n \vdash \bot$

You can always add/remove TRUE ($\top$)to/from the premises or FALSE ($\bot$) to/from the conclusions without changing the meaning of the sequent.

Why? Hint: Indentity laws

PVS commands: (FLATTEN)

(FLATTEN) eliminates $\wedge$ in the premises (by $\wedge e$) and $\vee$ in the conclusions (by $\vee i_1, \vee i_2$):

$$\begin{array}{|c} \phi_1 \wedge \phi_2 \\ \vdots \\ \hline \psi_1 \vee \psi_2 \\ \vdots \end{array} \quad \text{becomes} \quad \begin{array}{|c} \phi_1 \\ \phi_2 \\ \vdots \\ \hline \psi_1 \\ \psi_2 \\ \vdots \end{array}$$

(FLATTEN) also eliminates $\rightarrow$ in the conclusions:

$$\begin{array}{|c} \phi \\ \hline \psi_1 \rightarrow \psi_2 \end{array} \text{ becomes } \begin{array}{|c} \phi \\ \psi_1 \\ \hline \psi_2 \end{array}$$

Why?

PVS commands: (FLATTEN)

(FLATTEN) eliminates negations:

$$\begin{array}{|c} \phi_1 \\ \hline \neg\psi \\ \psi_1 \\ \psi_2 \end{array} \quad \text{becomes} \quad \begin{array}{|c} \phi_1 \\ \psi \\ \hline \psi_1 \\ \psi_2 \end{array}$$

Why? $\phi_1 \vdash \neg\psi \vee \psi_1 \vee \psi_2$ iff $\phi_1 \vdash \psi \rightarrow (\psi_1 \vee \psi_2)$ iff $\phi_1, \psi \vdash \psi_1 \vee \psi_2$

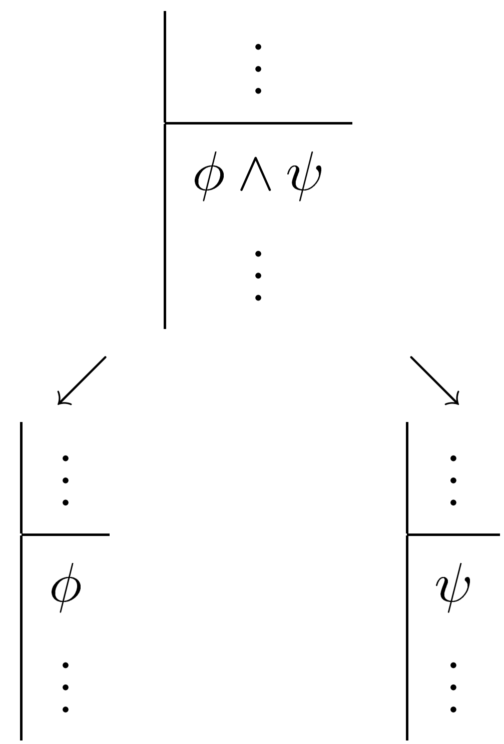
PVS commands: (FLATTEN)

Similarly $\phi_1, \neg\phi \vdash \psi_1 \vee \psi_2$ iff $\phi_1 \vdash \neg\phi \rightarrow \psi_1 \vee \psi_2$ iff $\phi_1 \vdash \neg\neg\phi \vee (\psi_1 \vee \psi_2)$ iff $\phi_1 \vdash \phi \vee (\psi_1 \vee \psi_2)$

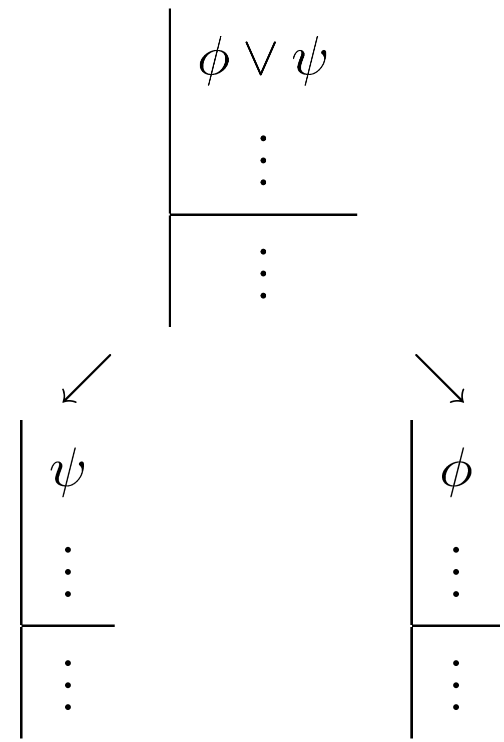
$$\begin{array}{|c} \phi_1 \\ \neg\phi \\ \hline \psi_1 \\ \psi_2 \end{array} \text{ becomes } \begin{array}{|c} \phi_1 \\ \hline \phi \\ \psi_1 \\ \psi_2 \end{array}$$

PVS commands: (SPLIT)

(SPLIT) uses "AND introduction" ($\wedge i$) to "split" a $\wedge$ in the conclusions into two subproofs (i.e. $\Gamma \vdash \phi \wedge \psi$ iff $\Gamma \vdash \phi$ and $\Gamma \vdash \psi$)



(SPLIT) uses "OR elimination" ($\vee e$) to "split" a $\vee$ in the premises into two subproofs (i.e. $\Gamma, \phi \vee \psi \vdash r$ iff $\Gamma, \phi \vdash r$ and $\Gamma, \psi \vdash r$)



(SPLIT) also splits $\leftrightarrow$ in the conclusions since:

$$(\phi \leftrightarrow \psi) \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$$

and splits $\rightarrow$ in the premises (why?).

.

PVS commands: (BDDSIMP)

The BDDSIMP command, in effect,

1. creates the truth table for the characteristic formula of the sequent. If it is a tautology the proof is done because

$$\models \phi \rightarrow \psi \text{ iff } \vdash \phi \rightarrow \psi \text{ iff } \phi \vdash \psi$$

   (take $\phi : \phi_1 \wedge \ldots \phi_n$ and $\psi : \psi_1 \vee \ldots \psi_m$). Otherwise BDDSIMP

2. obtains the CNF representation,
3. simplifies it with the help of the distributive law, and
4. applies the Rule of Adjunction to split the sequent into one sub-proof for each uninterupted sequence of disjuncts and flattens all negations.

NOTE: BDDs - (ordered) Binary Decision Diagrams, are type of data structure representing a formula that can be algorithmically reduced to a canonical representation.

(BDDSIMP) Example

Applying (BDDSIMP) to sequent $\vdash p \to q \wedge r$:

1. Create Truth Table for $p \to q \wedge r$.
2. Get DNF for $\neg(p \to q \wedge r)$ then negate and "De Morgan it to death" to get (full) CNF or write down CNF directly:

$$(\neg p \vee q \vee r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$$

3. Simplify to: $(\neg p \vee q) \wedge (\neg p \vee r)$
4. Split to get $\dfrac{\quad}{\neg p \vee q}$ and $\dfrac{\quad}{\neg p \vee r}$ then

   flatten to $\dfrac{p}{q}$ and $\dfrac{p}{r}$

Fill in details of $\vdash p \rightarrow q \wedge r$ (BDDSIMP) example.

Checking Validity of Arguments in PVS

By Theorems on Soundness and Completeness $\phi_1, \phi_2, \ldots \phi_n \models \psi$ iff $\models \phi_1 \wedge \ldots \wedge \phi_n \rightarrow \psi$

i.e. $\phi_1 \wedge \ldots \wedge \phi_n \rightarrow \psi$ is a tautology.

Therefore to check if $\phi_1, \ldots, \phi_n$ are a valid argument for ψ, use PVS to prove the theorem:

`V1:  THEOREM` $\phi_1 \& \ldots \& \phi_n$ `IMPLIES` ψ

Checking Consistency of Premises in PVS

The set of premises $\phi_1, \ldots, \phi_n$ is inconsistent iff $\phi_1, \ldots, \phi_n \vdash \psi \wedge \neg\psi$ for some ψ iff $\phi_1, \ldots, \phi_n \vdash \bot$

But then by the deduction theorem ($\rightarrow i$):

$$\vdash \quad \phi_1 \rightarrow (\phi_2 \rightarrow (\phi_3 \rightarrow (\ldots \rightarrow (\phi_n \rightarrow \bot) \ldots)))$$

iff

$$\vdash \quad \phi_1 \wedge \phi_2 \wedge \phi_3 \ldots \wedge \phi_n \rightarrow \bot$$

iff

$$\vdash \quad \neg(\phi_1 \wedge \phi_2 \wedge \phi_3 \ldots \wedge \phi_n)$$

Therefore propositional premises $\phi_1, \ldots, \phi_n$ are inconsistent iff you can prove the PVS theorem:

`V1:` `THEOREM` $\phi_1 \& \ldots \& \phi_n$ `IMPLIES` $FALSE$ or equivalently

`V2:` `THEOREM` $\neg(\phi_1 \& \ldots \& \phi_n)$

Unprovable Sequents & Counter Examples

Consider the following example:

Use PVS to check if the argument following argument is valid & find a counter example if it is not:

$$q \to m \vee v, m, v \to q \stackrel{?}{\models} q$$

```
E1 :  THEOREM (q IMPLIES m OR v) & m &
        (v IMPLIES q) IMPLIES q
```

Trying (BDDSIMP) gives unprovable sequent.

```
{-1}    m
  |-------
{1}    q
{2}    v
```

which has characteristic formula $m \to (q \vee v)$. This formula is false when $m = T$ and $q = v = F$. Check that this provides a counter example showing the argument is not valid.

Example: Understanding PVS

Use PVS to show:

$$\vdash ((p \to q) \to q) \to ((q \to p) \to p)$$

Explain the proof steps.

Solution: In PVS file we have

```
p,q:bool
a2i:theorem ((p=>q)=>q)=>((q=>p)=> p)
```

Invoking the prover:

```
  |-------
{1}   ((p => q) => q) => ((q => p) => p)


Rule? (FLATTEN)
```

```
Applying disjunctive simplification to
flatten sequent, this simplifies to:
a2i :

{-1}  ((p => q) => q)
{-2}  (q => p)
  |-------
{1}   p
```

Note that if

$$(p \to q) \to q), (q \to p) \quad \vdash \quad p$$

Then by $\to i$

$$(p \to q) \to q) \quad \vdash \quad (q \to p) \to p$$

And also by $\to i$

$$\vdash \quad (p \to q) \to q) \to ((q \to p) \to p)$$

Thus it suffices to show

$$(p \to q) \to q), (q \to p) \quad \vdash \quad p$$

```
a2i :
{-1}  ((p => q) => q)
{-2}  (q => p)
  |-------
{1}   p

Rule? (SPLIT -1)
Splitting conjunctions,
this yields  2 subgoals:
a2i.1 :
{-1}  q
[-2]  (q => p)
  |-------
[1]   p
Rule? (SPLIT)
```

```
Splitting conjunctions,
this yields  2 subgoals:
a2i.1.1 :

{-1}  p
[-2]  q
  |-------
[1]   p

which is trivially true.
This completes the proof of a2i.1.1.

a2i.1.2 :

[-1]  q
  |-------
```

```
{1}   q
[2]   p
```

which is trivially true.

This completes the proof of a2i.1.2.

This completes the proof of a2i.1.

```
a2i.2 :

[-1]  (q => p)
  |-------
{1}   (p => q)
[2]   p

Rule? (split -1)
Splitting conjunctions,
this yields  2 subgoals:
a2i.2.1 :

{-1}  p
  |-------
[1]   (p => q)
[2]   p
```

```
which is trivially true.

This completes the proof of a2i.2.1.

This completes the proof of a2i.1.
a2i.2.2 :

  |-------
{1}   q
[2]   (p => q)
[3]   p

Rule? (flatten)
Applying disjunctive simplification
to flatten sequent.
```

This completes the proof of a2i.2.2.

This completes the proof of a2i.2.

Q.E.D.

Example: Laplante Real-Time Systems Design and Analysis (3rd ed)

Consider the following excerpt from the Software Requirements Specification for the nuclear monitoring system.

1.1 If interrupt A arrives, then task B stops executing.

1.2 Task A begins executing upon arrival of interrupt A.

1.3 Either Task A is executing and Task B is not, or Task B is executing and Task A is not, or both are not executing.

These requirements can be formalized by rewriting each in terms of their component propositions, namely:

p: interrupt A arrives

q: task B is executing

r: task A is executing

Rewriting the requirements in proposition logic yields:

1.1 $p \rightarrow \neg q$

1.2 $p \rightarrow r$

1.3 $(r \wedge \neg q) \vee (r \wedge \neg r) \vee (\neg q \wedge \neg r)$

Note that **1.3** is semantically equivalent to $\neg(q \wedge r)$.

We'll use this shorter version to check if the requirements are inconsistent. i.e.

$$p \rightarrow \neg q, p \rightarrow r, \neg(q \wedge r) \vdash \bot$$

If they are inconsistent, then no program exists that satisfies them all. Conversely, if the requirements are consistent, we need to find a counter example showing:

$$p \rightarrow \neg q, p \rightarrow r, \neg(q \wedge r) \not\models \bot$$

You can do this by hand, but in PVS we could use:

```
demo04                        : THEORY
  BEGIN
  p, q, r: bool

  Laplante: Theorem
(p=> NOT q) & (p =>r) & NOT(q & r) => FALSE
  END demo04
```

Invoking the prover and running the `(BDDSIMP)` command results in two unprovable sequents.

```
Laplante.1 :

{-1}  r
  |-------
{1}   q
```

```
and

Laplante.2 :

  |-------
{1}   p
{2}   r
```

The first has characteristic eqn. $r \rightarrow q$ which gives counter example $q = F$ and $r = T$. Checking the truth table we have a counter example:

p	q	r	$p \rightarrow \neg q$	$p \rightarrow r$	$\neg(q \wedge r)$	$\perp$
F	F	T	T	T	T	F