Formalised Cut-elimination for the Display Calculus of Relation Algebras

Jeremy E. Dawson and Rajeev Goré

Automated Reasoning Group
Computer Sciences Laboratory
Res. Sch. of Inf. Sci. and Eng.
Institute of Advanced Studies
Australian National University

jeremy@arp.anu.edu.au
arp.anu.edu/~jeremy

Formal Methods Group
Dept. of Computer Science
Fac. of Eng. and Inf. Tech.
The Faculties
Australian National University

rpg@arp.anu.edu.au
arp.anu.edu.au/~rpg

Goré supported by an Australian Research Council Queen Elizabeth II Fellowship.
Dawson supported by an Australian Research Council Large Research Grant.
Semantics and Syntax of Logic of Relation Algebras

RA: Algebra of binary relations (sets of ordered pairs) over underlying set $U$

Atomic constants: $c ::= \bot | \top | 1 | 0$  
$(\emptyset | U \times U | \text{identity} | \text{diversity})$

Atomic formulae: $p ::= p_0 | p_1 | p_2 | \cdots$  
(relation symbols)

Formulae: $A ::= c | p | \neg A | A \land A | A \lor A | \sim A | A \circ A | A + A$

relative complement  $\cap$  $\cup$  converse composition

$\sim A := \{(y, x) | (x, y) \in A\}$

$A \circ B := \{(x, z) | \exists y \in U. (x, y) \in A \land (y, z) \in B\}$
Display Calculus $\delta_{RA}$ for Relation Algebra

**RA:** Algebra of binary relations (sets of ordered pairs) over underlying set $U$

**Atomic constants:** $c ::= \bot \mid \top \mid 1 \mid 0$  
$(\emptyset \mid U \times U \mid \text{identity} \mid \text{diversity})$

**Atomic formulae:** $p ::= p_0 \mid p_1 \mid p_2 \mid \cdots$ (relation symbols)

**Formulae:** $A ::= c \mid p \mid \neg A \mid A \land A \mid A \lor A \mid \sim A \mid A \circ A \mid A + A$

relative complement  $\cap$  $\cup$  converse composition

**Structure:** $X ::= A \mid I \mid *X \mid X \times X \mid E \mid \bullet X \mid X ; X$

**Sequent:** $X \vdash Y$ where $X$ is the *antecedent* and $Y$ is the *succedent*

**Sequent rule:**

$\frac{X_1 \vdash Y_1 \cdots X_n \vdash X_n}{X \vdash Y}$

premises  conclusion

---

Australian National University  2003  3
Display Calculus $\delta_{RA}$: Display Postulates

\[
\begin{align*}
&\frac{*X \vdash Y}{*Y \vdash X} & \frac{X \vdash *Y}{Y \vdash *X} & \frac{* * X \vdash Y}{X \vdash Y} & \frac{\bullet X \vdash Y}{X \vdash \bullet Y} & \frac{\bullet \bullet X \vdash Y}{X \vdash Y} \\
&\frac{X \vdash Z ; \bullet * Y}{X ; Y \vdash Z} & \frac{Z ; \bullet * Y \vdash X}{Z \vdash X ; Y} & \frac{X \vdash Z , * Y}{X , Y \vdash Z} & \frac{Z , * Y \vdash X}{Z \vdash X , Y} & \frac{\bullet \bullet X \vdash Y}{Y \vdash \bullet X , Z} & \frac{\bullet X , Z \vdash Y}{* X , Z \vdash Y}
\end{align*}
\]

**Intuition:** Allow us to unravel complex structures: think of high school algebra where we solved $x^2 - 4 = 0$ by “making $x$ the subject”

**Invertible rules:** Reversible, merely bookkeeping device
## Display Calculus $\delta RA$: Display Postulates

$$
\begin{align*}
* X \vdash Y & \quad X \vdash * Y & \quad * * X \vdash Y & \quad \bullet X \vdash Y & \quad \bullet \bullet X \vdash Y \\
* Y \vdash X & \quad Y \vdash * X & \quad X \vdash Y & \quad X \vdash \bullet Y & \quad X \vdash Y \\
X \vdash Z ; * \bullet Y & \quad Z ; * \bullet Y \vdash X & \quad X \vdash Z , * Y & \quad Z , * Y \vdash X \\
X ; Y \vdash Z & \quad Z \vdash X ; Y & \quad X , Y \vdash Z & \quad Z \vdash X , Y \\
Y \vdash * \bullet X ; Z & \quad * \bullet X ; Z \vdash Y & \quad Y \vdash * X , Z & \quad * X , Z \vdash Y
\end{align*}
$$

**Theorem (Belnap82):** For every sequent $X \vdash Y$ and every substructure $Z$ of $X \vdash Y$, there is a structurally equivalent sequent $Z \vdash Y'$ or $X' \vdash Z$ that has $Z$ displayed as the whole of its antecedent or succedent.
Theorem (Belnap82): For every sequent $X \vdash Y$ and every substructure $Z$ of $X \vdash Y$, there is a structurally equivalent sequent $Z \vdash Y'$ or $X' \vdash Z$ that has $Z$ displayed as the whole of its antecedent or succedent.

\[
\begin{align*}
* X & \vdash Y & X & \vdash * Y & * * X & \vdash Y & \bullet X & \vdash Y & \bullet \bullet X & \vdash Y \\
* Y & \vdash X & Y & \vdash * X & X & \vdash Y & X & \vdash \bullet Y & X & \vdash Y \\
X & \vdash Z ; * \bullet Y & Z ; * \bullet Y & \vdash X & X & \vdash Z , * Y & Z , * Y & \vdash X \\
X ; Y & \vdash Z & Z & \vdash X ; Y & X , Y & \vdash Z & Z & \vdash X , Y \\
Y & \vdash * \bullet X ; Z & * \bullet X ; Z & \vdash Y & Y & \vdash * X , Z & * X , Z & \vdash Y \\
\end{align*}
\]

\[
\begin{align*}
*p_3 & \vdash p_0 ; (p_0 \circ p_1) & *p_3 ; * \bullet (p_0 \circ p_1) & \vdash p_0 & * \bullet p_0 ; p_3 & \vdash p_0 \circ p_1 & *p_3 & \vdash p_0 ; (p_0 \circ p_1) \\
*(p_0 ; (p_0 \circ p_1)) & \vdash p_3 & *p_3 & \vdash p_0 ; (p_0 \circ p_1) & \\
\end{align*}
\]
Display Calculus $\delta RA$: Logical Rules

➤ Left and right introduction rules for every logical connective

➤ (id) $p \vdash p$

➤ (cut) $\frac{X \vdash A \quad A \vdash Y}{X \vdash Y}$

➤ $(\vdash \vdash) \frac{A \vdash X \quad B \vdash Y}{A \lor B \vdash X ; Y}$
decoding/rewrite

➤ $(\vdash \vdash) \frac{Z \vdash A ; B}{Z \vdash A + B}$

➤ $(\land \vdash) \frac{A , B \vdash Z}{A \land B \vdash Z}$
rewrite/decoding

➤ $(\vdash \land) \frac{X \vdash A \quad Y \vdash B}{X , Y \vdash A \land B}$

➤ Introduced formula is always displayed

Antecedent Part $\top \quad \neg \quad \land \quad \bot$

➤ Gentzen overloading:

Connective $I \quad * \quad , \quad E \quad \bullet \quad ;$

Succedent Part $\bot \quad \neg \quad \lor \quad 0 \quad \supset \quad \vdash$

➤ Example: $\ast(p_0 ; p_1) \vdash \ast \bullet (p_1 , p_3)$
; is $+$ and , is $\land$
### Display Logic $\delta_{RA}$: Structural Rules

A *structural* rule contains no logical connectives and no formula variables.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(A; \vdash)$</td>
<td>$X; (Y; Z) \vdash W$</td>
<td>$(X; Y); Z \vdash W$</td>
</tr>
<tr>
<td>$(\vdash A)$</td>
<td>$W \vdash X, (Y, Z)$</td>
<td>$W \vdash (X, Y), Z$</td>
</tr>
<tr>
<td>$(\vdash P)$</td>
<td>$Z \vdash Y, X$</td>
<td>$Z \vdash X, Y$</td>
</tr>
<tr>
<td>$(I^\perp \vdash)$</td>
<td>$X, I \vdash Y$</td>
<td>$X \vdash Y$</td>
</tr>
<tr>
<td>$(\vdash E^\perp)$</td>
<td>$X \vdash E; Y$</td>
<td>$X \vdash Y$</td>
</tr>
<tr>
<td>$(\vdash \text{•E})$</td>
<td>$X \vdash \text{•E}$</td>
<td>$X \vdash \text{•E}$</td>
</tr>
<tr>
<td>$(\bullet I \vdash)$</td>
<td>$I \vdash X$</td>
<td>$I \vdash X$</td>
</tr>
<tr>
<td>$(M \vdash)$</td>
<td>$X \vdash Z$</td>
<td>$X, Y \vdash Z$</td>
</tr>
<tr>
<td>$(\vdash C)$</td>
<td>$Z \vdash X, X$</td>
<td>$Z \vdash X$</td>
</tr>
<tr>
<td>$(\text{tag})$</td>
<td>$X; Y \vdash Z$</td>
<td>$\bullet Y; \bullet X \vdash \bullet Z$</td>
</tr>
</tbody>
</table>

Note: no (M) (P) or (C) rules for $\circ/\perp$; since $\circ/\perp$ are not classical connectives.
Logical Variation Via Structural Rules (modularity)

➤ An equation $A \leq B$ is primitive iff $A$ and $B$ are built from relational variables and the constants $\top$ and $1$ with the help of $\sim$, $\land$, $\lor$ and $\circ$ only, and if no relational variable appears in $A$ more than once (Kracht96).

➤ Purely syntactic class of formulae
Logical Variation Via Structural Rules (modularity)

An equation $A \leq B$ is **primitive** iff $A$ and $B$ are built from relational variables and the constants $\top$ and $1$ with the help of $\sim$, $\land$, $\lor$ and $\circ$ only, and if no relational variable appears in $A$ more than once (Kracht96).

Examples: from modal logic

- Reflexivity
  - $\mathbf{T}$: $1 \leq A$
- Seriality
  - $D$: $A \circ \top = \top$
- Symmetry
  - $B$: $A = \sim A$
- Transitivity
  - $4$: $A \circ B \leq A \lor B$
Logical Variation Via Structural Rules (modularity)

➤ An equation $A \leq B$ is **primitive** iff $A$ and $B$ are built from relational variables and the constants $\top$ and $1$ with the help of $\neg$, $\land$, $\lor$ and $\circ$ only, and if no relational variable appears in $A$ more than once (Kracht96).

reflexivity   seriality        symmetry   transitivity  
$(T) \quad 1 \leq A$   $(D) \quad A \circ \top = \top$  $(B) \quad A \equiv \neg A$  $(4) \quad A \circ B \leq A \lor B$

$(T) \quad \ldfrac{X \vdash Z}{E \vdash Z}$  $(D) \quad \ldfrac{I \vdash Z}{X \land I \vdash Z}$  $(B) \quad \ldfrac{\bullet X \vdash Z}{X \vdash Z}$  $(4) \quad \ldfrac{X_1 \vdash Z \quad X_2 \vdash Z}{X_1 \land X_2 \vdash Z}$

➤ Kracht shows how to convert each primitive equation into a purely structural rule
Logical Variation Via Structural Rules (modularity)

- An equation $A \leq B$ is **primitive** iff $A$ and $B$ are built from relational variables and the constants $\top$ and $1$ with the help of $\sim$, $\wedge$, $\lor$ and $\circ$ only, and if no relational variable appears in $A$ more than once (Kracht96).

<table>
<thead>
<tr>
<th>reflexivity</th>
<th>seriality</th>
<th>symmetry</th>
<th>transitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(T)$ $1 \leq A$</td>
<td>$(D)$ $A \circ \top = \top$</td>
<td>$(B)$ $A = \sim A$</td>
<td>$(4)$ $A \circ B \leq A \lor B$</td>
</tr>
<tr>
<td>$(T)$ $\frac{X \vdash Z}{E \vdash Z}$</td>
<td>$(D)$ $\frac{I \vdash Z}{X; I \vdash Z}$</td>
<td>$(B)$ $\frac{\bullet X \vdash Z}{X \vdash Z}$</td>
<td>$(4)$ $\frac{X_1 \vdash Z \quad X_2 \vdash Z}{X_1; X_2 \vdash Z}$</td>
</tr>
</tbody>
</table>

- Theorem [Belnap 82]: Any display calculus whose rules obey eight conditions given by Belnap enjoys cut-elimination. If $X \vdash Y$ is derivable, then it is derivable without the cut rule.

- Theorem: Using results of Kracht 96 we can construct a *cut-free* display calculus which is sound and complete for any primitive extension of RA.
Formalising $\delta^R\mathcal{A}$ in Isabelle/HOL

**Formulae:** $A ::= \ c \mid \ p \mid \neg A \mid A \land A \mid A \lor A \mid \Leftrightarrow A \mid A \circ A \mid A + A$

datatype formula = Btimes formula formula ("_ && _" [68,68] 68)  
| Rtimes formula formula ("_ oo _" [68,68] 68)  
| Bplus formula formula ("_ v _" [64,64] 63)  
| Rplus formula formula ("_ ++ _" [64,64] 63)  
| Bneg formula ("--" [70] 70)  
| Rneg formula ("_ ^" [75] 75)  
| Btrue ("T")  
| Bfalse("F")  
| Rtrue ("r1")  
| Rfalse("r0")  
| PP string  
| FV string
Formalising $\delta_R^A$ in Isabelle/HOL

Structure: \[ X ::= A | I | \ast X | X \cdot X | E | \bullet X | X ; X \]

datatype structr = Comma structr structr ,
                  | SemiC structr structr ;
                  | Star structr *
                  | Blob structr @
                  | I $I$
                  | E $E$
                  | Structform formula
                  | SV string $'X'$

datatype sequent = Sequent structr structr |- $'X'$ $'A'$

$'X'$ |- $'A'$ is syntactic sugar for

Sequent (SV $'X'$) (Structform (FV $'A'$))
A Deep Embedding of Rules

\[ \frac{X_1 \vdash Y_1 \cdots X_n \vdash X_n}{X \vdash Y} \]
\[ X \vdash Y ; Z \]
\[ * \cdot Y ; X \vdash Z \]

```haskell
datatype rule = Rule (sequent list) sequent
| Bidi sequent sequent
| InvBidi sequent sequent
```

bs1 "bs1 == Bidi ($"'X'" |− $"'Y'" ; $"'Z'")
( *@$"'Y'" ; $"'X'" |− $"'Z'" )"

Deep embedding because of the "'X'". Shallow embedding would use \( ?X \).

This means we must handle substitutions explicitly.

\textit{rls} is the set of rules of \( \delta{RA} \)
Formalising Derivations

datatype dertree = Der sequent rule (dertree list)
   | Unf sequent

True derivation has leaves \[ \text{Der} (\text{PP} \ 'p' \ |- \ \text{PP} \ 'p') \ \text{id} \ [] \]

A derived rule will have some Unfinished leaf sequents

Example:

\[
\begin{align*}
A \vdash p & \quad A \vdash A \\
\hline
A, A \vdash p \land A
\end{align*}
\]

\[\text{Der (''A'' |- PP ''p'' && ''A'') cA} \]

\[\text{[Der (''A'', ''A'' |- PP ''p'' && ''A'') and} \]

\[\text{[Unf (''A'' |- PP ''p''),} \]

\[\text{Der (''A'' |- ''A'') idf []]} \]

Explain idf in a couple of slides
Reasoning About Derivations

datatype dertree = Der sequent rule (dertree list)  
               | Unf sequent

wfb :: "dertree => bool" returns true if the end-sequent of dertree and the end-sequents of its immediate children in dertree list are a well-formed instance of rule

allDT :: "(dertree => bool) => dertree => bool" returns true if every node of dertree satisfies the given boolean valued function

allDT wfb returns true if every rule application in dertree is well formed

allDT (frb rules) returns true if every rule application in dertree is a member of set rules
Formalising Derivability

IsDerivableR ::"rule set => [sequent set,sequent] => bool"

**Definition 1**  
IsDerivableR rules prems concl holds iff there exists a derivation tree dt which uses only rules from set rules, is well-formed, has conclusion concl, and has premisses from set prems.

"IsDerivableR rules prems concl == (EX dt. allDT (frb rules) dt & allDT wfb dt & conclDT dt = concl & set (premsDT dt) <= prems)"

**Lemma 1**  
The sequent PP "p" |- PP "p" is derivable for all atoms p

Proof: Let dt be Der (PP "p" |- PP "p") id [] in IsDerivableR rls {} (PP "p" |- PP "p")
Results About Derivability

Our main result about derivability is a recursive characterisation of derivability.

**Theorem 1** A conclusion `concl` is derivable from premisses `prems` using rules `rules` iff either `concl` is one of `prems`, or there exists an instantiated rule obtained from `rules` and the conclusion of rule is `concl` and the premisses of rule are themselves derivable from `prems` using `rules`.

```
  (?concl : ?prems
   | (EX rule. ruleFromSet rule ?rules
     & conclRule rule = ?concl
     & (ALL p : set (premsRule rule).
       IsDerivableR ?rules ?prems p))
  )" : thm
```
Formalising Derivable Rules

"IsDerivable rules (Rule prems concl) =
    IsDerivableR rules (set prems) concl"

Lemma 2 (idf) The sequent ‘’A’’ |- ‘’A’’ is derivable for all formulae
A that contain no formula variables

Add derived rule idf to rls:

    idf == Rule [] (‘’A’’ |- ‘’A’’)

Der (‘’A’’ |- ‘’A’’) idf [] stands for a complete derivation which
uses the derived rule idf that A ⊢ A is derivable

Unf (‘’A’’ |- ‘’A’’) stands for an incomplete derivation with unfinished
premiss A ⊢ A
Reducing a Left-and-Right Principal Cut

➤ A cut on $A$ is *left-[right-]principal* if the left [right] inference immediately above the cut introduces $A$ via a logical rule, written $(\vdash A) [(A \vdash)]$

$$
\frac{\prod_L (\vdash A) \quad \prod_R (A \vdash)}{X \vdash A \quad A \vdash Y}
\frac{}{X \vdash Y}
$$

➤ C8: All left-and-right principal cuts can be replaced by cuts on strictly smaller subformulae of $A$

➤ Base Case: (cut) $p \vdash p \quad p \vdash p$ reduced to just $p \vdash p$

➤ Similar cases for $\top, 1, \bot, 0$
Example of C8 condition

\[\frac{\Pi_{ZAB}}{Z \vdash A, B} \quad \Pi_{BY}\]
\[\frac{A \vdash X \quad B \vdash Y}{\vdash X, Y} \quad (\text{cut})\]

\[\frac{A \vdash X \quad B \vdash Y}{\vdash A \lor B} \quad (\text{cut})\]
\[\frac{Z \vdash A \lor B}{\vdash Z, \neg X \lor Y} \quad (\text{cut})\]

Principal cut on formula \(A \lor B\)

Transformed derivation uses cuts only on \(A\) and \(B\)

Must exhibit and hard-wire such transformations for every logical connective

So if cut is \textit{not} left-and-right principal then we try to make it so ...
Making a cut left-principal

➤ Given a derivation ending in a non left-and-right principal cut on formula $A$, we trace the ancestor occurrences of this $A$ up $\Pi_L$:

\[
\begin{array}{c}
\Pi_L \\
X \vdash A \\
\Pi_R \\
A \vdash Y \\
\hline
X \vdash Y
\end{array}
\]

➤ an ancestor of $A$ is *principal* if introduced by logical rule ($\vdash A$)

➤ all other ancestors of $A$ are *parametric* ancestors (including those “introduced” by weakening)

➤ there must be at least one introduction of $A$ in every branch

➤ not all occurrences of $A$ need be ancestors of this occurrence of $A$
Making a cut left-principal

➤ Given a derivation ending in a non left-and-right principal cut on formula $A$, we trace all ancestor occurrences of this $A$ up $\Pi_L$:

➤ Let $\Pi[A := Y]$ mean “in $\Pi$, replace all parametric ancestors of $A$ with $Y$”

➤ Suppose $\Pi_L$ does not branch at all for simplicity

\[
\begin{align*}
\Pi_L^1 & \quad (\vdash A) \\
V & \vdash A \\
\Pi_L^2 & \quad (\pi) \\
X & \vdash A \\
\Pi_L & \quad (\rho) \\
X & \vdash A \\
\Pi & \quad (\tau) \\
A & \vdash Y \\
\end{align*}
\]

➤ New cut is left-principal
Making a cut left-principal

• Base case of induction, $\Pi^1_L$ is empty, $A$ introduced by $A \vdash A$. No new cuts needed

$$
\begin{align*}
A \vdash A & \quad (\pi) \\
\Pi^2_L & \quad (\pi) \\
X \vdash A & \quad (\rho) \\
\Pi_R & \quad (\rho) \\
\frac{\Pi^2_L \quad \Pi_R}{X \vdash Y} & \quad (cut) \\
\frac{A \vdash Y}{X \vdash Y} & \quad (\rho) \\
\frac{\Pi^2_L \vdash Y}{X \vdash Y} & \quad (\rho)
\end{align*}
$$

• Case for "introduction" by weakening, no new cuts needed

$$
\begin{align*}
\Pi^1_L & \quad (Wk \ Z(A)) \\
\Pi^2_L & \quad (\pi) \\
X \vdash A & \quad (\rho) \\
\Pi_R & \quad (\rho) \\
\frac{\Pi^2_L \quad \Pi_R}{X \vdash Y} & \quad (cut) \\
\frac{A \vdash Y}{X \vdash Y} & \quad (\rho) \\
\frac{\Pi^2_L \vdash Y}{X \vdash Y} & \quad (\rho)
\end{align*}
$$
Cut-Admissibility (Weak Normalisation)

Show that the cut rule \( \frac{X \vdash A \quad A \vdash Y}{X \vdash Y} \) is admissible.

Assume: we have cut-free derivations of \( X \vdash A \) and \( A \vdash Y \).

Show: that there is a cut-free derivation of \( X \vdash Y \).

In general, cut-elimination proceeds by repeatedly eliminating an arbitrary top-most cut.

Must show that this procedure terminates.

Usual argument is a double induction on the size (degree) of the cut formula and the heights of the given derivations (grade).
Cut-Admissibility By Structural Induction

\texttt{elim} eliminates a single arbitrary top-most cut, by turning it into several left-principal cuts and using \texttt{elimAllLP} to eliminate them ... 

Our initial derivation contains only one bottom-most cut which is not left and right principal and we know how to turn it into multiple left principal cuts
Cut-Admissibility By Structural Induction

\texttt{elim} eliminates a single arbitrary top-most cut, by turning it into several left-principal cuts and using \texttt{elimAllLP} to eliminate them . . .

all new cuts are left principal

\texttt{elimAllLP} eliminates several left-principal cuts, by repeatedly using \texttt{elimLP} to eliminate the top-most remaining one . . .
Cut-Admissibility By Structural Induction

**elim** eliminates a single arbitrary top-most cut, by turning it into several left-principal cuts and using **elimAllLP** to eliminate them ...

**elimAllLP** eliminates several left-principal cuts, by repeatedly using **elimLP** to eliminate the top-most remaining one ...

**elimLP** eliminates a top-most left-principal cut, by turning it into several principal cuts and using **elimAllLRP** to eliminate them ...

all new cuts are left-and-right principal
Cut-Admissibility By Structural Induction

**elim** eliminates a single arbitrary top-most cut, by turning it into several left-principal cuts and using **elimAllLP** to eliminate them . . .

**elimAllLP** eliminates several left-principal cuts, by repeatedly using **elimLP** to eliminate the top-most remaining one . . .

**elimLP** eliminates a top-most left-principal cut, by turning it into several principal cuts and using **elimAllLRP** to eliminate them . . .

**elimAllLRP** eliminates several principal cuts, by repeatedly using **elimLRP** to eliminate the top-most remaining one . . .

**elimLRP** eliminates a top-most principal cut, by turning it into several cuts on *smaller* cut-formulae, and using **elimAll** to eliminate them . . .

all new cuts are on strictly smaller cut-formulae
Cut-Admissibility By Structural Induction

\texttt{elim} eliminates a single arbitrary top-most cut, by turning it into several left-principal cuts and using \texttt{elimAllLP} to eliminate them . . .

\texttt{elimAllLP} eliminates several left-principal cuts, by repeatedly using \texttt{elimLP} to eliminate the top-most remaining one . . .

\texttt{elimLP} eliminates a top-most left-principal cut, by turning it into several principal cuts and using \texttt{elimAllLRP} to eliminate them . . .

\texttt{elimAllLRP} eliminates several principal cuts, by repeatedly using \texttt{elimLRP} to eliminate the top-most remaining one . . .

\texttt{elimLRP} eliminates a top-most principal cut, by turning it into several cuts on smaller cut-formulae, and using \texttt{elimAll} to eliminate them . . .

\texttt{elimAll} eliminates several arbitrary cuts, by repeatedly using \texttt{elim} to eliminate the top-most remaining one . . .

\textit{termination}
Cut-Admissibility

" [ | canElim (cutOnFmls {?A}); canElim (cutOnFmls {?B}) | ]
  ==> canElim (cutOnFmls {?A v ?B})"

A proof by induction on the structure of a formula gives

canElimFml = "canElim (cutOnFmls {?fml})"

(“you can eliminate one cut on any given formula fml”),
canElimAny = "canElim (cutOnFmls UNIV)",

(“you can eliminate one cut whatever the cut-formula”), and, using elimFmls,
canElimAll = "canElimAll (cutOnFmls UNIV)"

Finally we convert canElimAll into

cutElim = "IsDerivableR rls {} ?concl ==> 
      IsDerivableR (rls - {cut}) {} ?concl"
Strong Normalisation

Cut-admissibility proceeds by eliminating the top-most cut

Strong normalisation: define reduction steps, eliminate any cut that can be reduced, and prove that this cut-elimination procedure terminates:

1. $\Pi_n < \Pi_{n-1} < \cdots < \Pi_2 < \Pi_0$

2. $\Pi_n$ is cut-free

3. $n$ is finite

4. weak-normalisation (attacking top most cut) is an instance of this method

Usually define reductions as either a parametric or principal move, with proviso that we must not cross a cut in a parametric move
Why Formalise (in Isabelle) ?

Strong-normalisation proofs are notoriously difficult involving many cases.

We tried to formalise Heinrich Wansing’s proof of strong-normalisation for display logic, but got stuck at one particular case.
Why Formalise (in Isabelle)?

Strong-normalisation proofs are notoriously difficult involving many cases.

We tried to formalise Heinrich Wansing’s proof of strong-normalisation for display logic, but got stuck at one particular case.

Found a bug ... Wansing assumes that $\Pi^1_L$ is unchanged!

\[
\begin{align*}
\frac{\Pi^1_L}{V \vdash A} & \quad (\vdash A) \text{ } V \text{ is } (*A, X) \\
\frac{X \vdash A, A}{X \vdash A} & \quad (dp) \\
\frac{X \vdash A}{X \vdash A} & \quad (ctr) \\
\frac{\Pi^1_L}{X \vdash A} & \quad (cut) \\
\frac{\Pi^1_L}{A \vdash Y} & \quad (cut) \\
\frac{V \vdash Y}{\Pi^1_L[A := Y]} & \quad (\pi) \\
\frac{X \vdash Y}{X \vdash Y} & \quad (\rho)
\end{align*}
\]

Can construct counter-examples where his proof goes into loops: a given cut is reduced to itself ... his procedure will not terminate.
Defining Reductions

**Definition 2** Assuming a relation \( \text{cutReduces} \) (defined later), derivation tree \( \Pi_0 \) *reduces to derivation tree* \( \Pi_1 \) if either

(a) \( \Pi_0 \) cut\( \text{Reduces} \) to \( \Pi_1 \), or

(b) \( \Pi_0 \) and \( \Pi_1 \) are identical except that exactly one of the immediate subtrees of \( \Pi_0 \) reduces to the corresponding immediate subtree of \( \Pi_1 \).

\[
\text{reduces}_\text{Unf} = "\text{reduces} \ (\text{Unf} \ ?\text{seq}) \ ?\text{dtm} = \text{False}"
\]

\[
\text{reduces}_\text{Der} = "\text{reduces} \ (\text{Der} \ ?\text{seq} \ ?\text{rule} \ ?\text{dtl}) \ ?\text{dtm} = \n\ (\ (\exists \ \text{dtl}’. \ \text{onereduces} \ ?\text{dtl} \ \text{dtl}’ & ?\text{dtm} = \text{Der} \ ?\text{seq} \ ?\text{rule} \ \text{dtl}’ \n\ | \ \text{cutReduces} \ (\text{Der} \ ?\text{seq} \ ?\text{rule} \ ?\text{dtl}) \ ?\text{dtm} )"
\]

\[
\text{onereduces}_\text{Nil} = "\text{onereduces} \ [ ] \ ?\text{dtl}’ = \text{False}"
\]

\[
\text{onereduces}_\text{Cons} = "\text{onereduces} \ (?h \ # \ ?t) \ ?\text{dtl}’ = ( \ ?\text{dtl}’ \sim= [ ] \n\ & \ (\text{reduces} \ ?h \ (\text{hd} \ ?\text{dtl}’) \ & \ ?t = (\text{tl} \ ?\text{dtl}’) \n\ | \ ?h = (\text{hd} \ ?\text{dtl}’) \ & \ \text{onereduces} \ ?t \ (\text{tl} \ ?\text{dtl}’)) \)"
\]
Defining Strongly Normalising Derivation Trees

Definition 3  The set \( sn_set \) is the smallest set of derivation trees such that:

(a) if \( \Pi_0 \) cannot be reduced then \( \Pi_0 \in sn_set \)

(b) if every tree \( \Pi_1 \) to which \( \Pi_0 \) reduces is in \( sn_set \) then \( \Pi_0 \in sn_set \).

A derivation tree is strongly normalisable iff it is a member of \( sn_set \).

inductive "sn_set"
  intrs

Example, the rules \( \{0 \in S, n \in S \Rightarrow n + 2 \in S\} \) define \( S \) to be the set of even naturals, although the set of all naturals also satisfies the rules.
Various Binary Orderings

1. $\Pi_1 <_{\text{LRP}} \Pi_0$ if the bottom inferences of derivations $\Pi_1$ and $\Pi_0$ are both cut, and either
   
   a. the cut in $\Pi_1$ is left-principal or right-principal, and the cut in $\Pi_0$ is neither, or
   
   b. the cut in $\Pi_1$ is (left- and right-)principal, and the cut in $\Pi_0$ is not (left- and right-)principal.

2. $\Pi_1 <_{\text{cut}} \Pi_0$ if the bottom inferences of derivations $\Pi_1$ and $\Pi_0$ are both cut, and if either
   
   a. the size of the cut-formula of $\Pi_1$ is smaller than that of $\Pi_0$, or
   
   b. each derivation has the same cut-formula, and $\Pi_1 <_{\text{LRP}} \Pi_0$. 
Various Binary Orderings

1. \( \Pi_1 <_{sn1} \Pi_0 \) if \( \Pi_0 \) and \( \Pi_1 \) are the same except that one of the immediate subtrees of \( \Pi_0 \) is strongly normalisable and reduces to the corresponding immediate subtree of \( \Pi_1 \).

2. \( \Pi_1 <_{dt} \Pi_0 \) iff any of the following hold:
   a. the bottom inference of \( \Pi_1 \) is not cut, but that of \( \Pi_0 \) is
   b. \( \Pi_1 <_{cut} \Pi_0 \)
   c. \( \Pi_1 <_{sn1} \Pi_0 \).

**Theorem 2**  *The relations LRPorder, cutorder, sn1order, and dtorder are well-founded*

Despite notation, these relations are *not* reflexive, and some are *not* transitive.

Intuitively, \((\Pi_1, \Pi_0) \in dtorder\) means that \( \Pi_1 \) is closer to being cut-free (in some sense) than is \( \Pi_0 \).
Inheriting Strong Normalisation

We next define \texttt{snHered}, a property of derivation trees which indicates that a tree inherits strong normalisation from its immediate subtrees.

\textbf{Definition 4} \( \Pi \) satisfies \texttt{snHered} iff: if all the immediate subtrees of \( \Pi \) are strongly normalisable then \( \Pi \) is strongly normalisable.

\[
\text{snHered} \_\text{def} = "\text{snHered} \ ?dt = = \\
\ 
\text{set (nextUp} \ ?dt) \ <= \text{sn_set} \ \\
\ 
\text{-->} \ ?dt : \text{sn_set}" 
\]

\textbf{Lemma 3} A derivation tree \( \Pi \) is strongly normalisable iff every subtree of \( \Pi \) has the property \texttt{snHered}.

Proof: The “only if” part uses a clever instantiation of a theorem generated automatically by Isabelle from the inductive definition of \texttt{sn_set}.

The “if” part uses induction on the height of \( \Pi \) using the assumption that every subtree of \( \Pi \), including itself, has the property \texttt{snHered}.
Defining Cut-Reductions

**Definition 5** Two derivation trees $\Pi_0$ and $\Pi_1$ satisfy $nparRedP$ if every subtree $\Pi_1^s$ of $\Pi_1$ with a bottom inference (cut) satisfies either

(a) $\Pi_1^s$ is a proper subtree of $\Pi_0$, or

(b) $\Pi_1^s$ and $\Pi_0$ have the same cut-formula and $\Pi_1^s <_{LRP} \Pi_0$.

**Definition 6** Two derivation trees $\Pi_0$ and $\Pi_1$ satisfy $c8redP$ if the lowest rule of $\Pi_0$ is cut, and for each subtree $\Pi_1^s$ of $\Pi_1$ whose lowest rule is cut, either

(a) $\Pi_1^s$ is a proper subtree of $\Pi_0$, or

(b) the cut-formula of $\Pi_1^s$ is smaller than that of the lowest rule of $\Pi_0$. 
Defining Cut-Reduction and Hence Reduction

**Definition 7** The derivation tree $\Pi_0$ cut-reduces to $\Pi_1$ if the following hold simultaneously:

(a) $\Pi_0$ and $\Pi_1$ satisfy either $\text{nparRedP}$ (for a parametric reduction) or $\text{c8redP}$ (for a principal reduction)

(b) $\Pi_0$ and $\Pi_1$ have the same conclusion

(c) the bottom rule of $\Pi_0$ is cut

(d) $\Pi_0$ and $\Pi_1$ are not identical

(e) $\Pi_1$ does not consist solely of an unfinished leaf

(f) and $\Pi_0$ has at least one immediate subtree
**Strong-Normalisation**

**Lemma 4**  *For a given derivation* $\Pi_0$, *if all derivation trees* $\Pi' <_{dt} \Pi_0$ *have the property* $snHered$, *then so does* $\Pi_0$.

\[dth = "\text{ALL } dt'. (dt', ?dt) : dtorder --> snHered dt' \Rightarrow snHered ?dt" : thm\]

Proof: The machine-proof is quite complicated, and a careful examination of it highlights why the definition of $dtorder$ needs to be so complex.

**Theorem 3**  *Every derivation tree is strongly normalisable.*

\[all_snH = "snHered ?dt" : thm\]
\[all_sn = "strongNorm ?dt" : thm\]

Proof: By well-founded induction, it follows from Lemma 4 that every derivation tree satisfies $snHered$; the result follows from Lemma 3.
Where are we?

We have actually shown using Isabelle that a class of reductions which we have defined is strongly normalising. We need to show that this class of reductions contains the ones in which we are interested.
Dealing With Parametric Cuts

**Theorem 4**  Consider a parametric reduction of a cut which proceeds by transforming the left subtree (to change its conclusion from $X[A] \vdash A$ to $X[Y] \vdash Y$), using the function $mLP$. Assume that the transformation can be performed without traversing another cut. Also assume the cut is not already left-principal. Then the reduction satisfies the condition $nparRedP$.

**Theorem 5**  The same as Theorem 4, except with conclusion “Then the reduction satisfies the condition $cutReduces$”.

Analogous theorems $pRedRP2$, $pRedRP3$ guarantee that we can make a cut right-principal, and consequently another theorem which guarantees that we can make a cut (left and right) principal.

The associated functions also produce derivations that satisfy $cutReduces$.

Thus every parametric move corresponds to a reduction in the class of strongly normalising reductions.
Dealing With Principal Cuts

Theorem 6  Case for $\lor$: Given a valid derivation tree $dt$, assume that if the bottom rule of $dt$ is cut, then the cut is principal and its cut-formula is $A \lor B$. Then there is a valid derivation tree $dtn$ with the same conclusion as $dt$ such that $dt$ and $dtn$ satisfy $c8_{resfP}$ (since the original cut on $A \lor B$ is replaced with new cuts on its proper subformulae $A$ and $B$).

These principal reductions satisfy $c8_{redfsP}$ and hence $c8_{redP}$, and are in fact cut-reductions. These results (one for each logical connective) give

Theorem 7  If the bottom-most rule of a valid derivation tree $\Pi_0$ is a (left and right) principal cut, then there exists a valid derivation tree $\Pi_1$ with the same conclusion as $\Pi_0$, such that $\Pi_0$ cutReduces to $\Pi_1$.

Thus every principal move gives a reduction which is in the class of strongly normalising reductions.
Definition 8  A valid reduction is a reduction of tree $\Pi_0$ to tree $\Pi_1$, where $\Pi_1$ is a valid tree which has the same conclusion as $\Pi_0$.

Theorem 8  Parametric and principal moves give valid reductions.

Theorem 9  For any valid tree which contains a cut, there is available at least one valid reduction (reduce a top-most cut).

Corollary: Any sequence of principal and parametric moves starting from some derivation $\Pi$ containing cuts, eventually terminates with a cut-free derivation $\Pi^r$ that has the same conclusion as $\Pi$.

Theorem 10  If a sequent can be derived using rules $\text{rls}$, then it can be derived from those rules omitting cut.

Belnap’s Conditions and New C8

Appendix: Belnap’s Conditions.

For every sequent rule Belnap82, page 388 first defines the following notions: in an application $\text{Inf}$ of a sequent rule $\rho$, “constituents occurring as part of occurrences of structures assigned to structure-variables are defined to be parameters of $\text{Inf}$; all other constituents are defined as nonparametric, including those assigned to formula-variables. Constituents occupying similar positions in occurrences of structures assigned to the same structure-variable are defined as congruent in $\text{Inf}$”. The eight (actually seven) conditions shown below are from [Kracht96]:

(C1) Each formula which is a constituent of some premiss of a rule $\rho$ is a subformula of some formula in the conclusion of $\rho$.

(C2) Congruent parameters are occurrences of the same structure.
(C3) Each parameter is congruent to at most one constituent in the conclusion. Equivalently, no two constituents of the conclusion are congruent to each other.

(C4) Congruent parameters are either all antecedent parts or all succedent parts of their respective sequent.

(C5) If a formula is non-parametric in the conclusion of a rule $\rho$, it is either the entire antecedent, or the entire succedent. Such a formula is called a principal formula.

(C6/7) Each rule is closed under simultaneous substitution of arbitrary structures for congruent parameters.
(C8) If there are inference rules $\rho_1$ and $\rho_2$ with respective conclusions $X \vdash \varphi$ and $\varphi \vdash Y$ with $\varphi$ principal in both inferences (in the sense of C5), and if (cut) is applied to yield $X \vdash Y$ then, either $X \vdash Y$ is identical to $X \vdash \varphi$ or to $\varphi \vdash Y$; or it is possible to pass from the premisses of $\rho_1$ and $\rho_2$ to $X \vdash Y$ by means of inferences falling under (cut) where the cut-formula is always a proper subformula of $\varphi$. 