BETH'S TABLEAUX FOR RELEVANT LOGIC

J. F. PABION

1 Sequents—Countermodels For the concept of relevant-model-structure (r.m.s.) we* refer to [2]. Giving a r.m.s. \( \mathfrak{M} = \langle K, 0, R, * \rangle \) satisfying postulates \( P_1 - P_6 \) of [2], a "forcing relation" is defined between \( K \) and the set of atomic sentences defined in a given domain \( D \) such that:

\[ \alpha \models A \text{ and } R(0, \alpha, \beta) \Rightarrow \beta \models A. \]

Then the forcing relation is extended according to usual stipulations:

\[ \begin{align*}
\alpha &\models A \lor B \iff \alpha \models A \text{ or } \alpha \models B \\
\alpha &\models A \land B \iff \alpha \models A \text{ and } \alpha \models B \\
\alpha &\models A \rightarrow B \iff \forall \beta, \gamma, \beta \models A \text{ and } R(\alpha, \beta, \gamma) \Rightarrow \gamma \models B \\
\alpha &\models \neg A \iff \alpha^* \not\models A \\
\alpha &\models \forall x F[x] \iff \forall a \in D, \alpha \models F[a].
\end{align*} \]

A signed formula is an expression of one of the forms:

\[ +A \quad -A \]

where \( A \) is a formula. A sequent is a finite sequence of signed sentences.

A countermodel of a sequent is given by:

\[ \mathfrak{M} = \langle K, 0, R, * \rangle. \]

A member \( a \) of \( K \) such that:

If \( +A \) is in the sequent, then \( a \models A \)

If \( -A \) is in the sequent, then \( a \not\models A \).

\( \langle \mathfrak{M}, a \rangle \) is a normal countermodel if \( a = 0 \) and \( 0^* = 0 \). The sequent is said to be valid if it has no countermodel and normally valid if it has no normal countermodel.

*The author is grateful to professors Belnap, Dunn, and Meyer for having focalized his attention on typical mistakes contained in some tentative proofs.

Received May 10, 1977
In what follows, we shall describe a Beth's type method to verify normal validity. It would not be difficult to adapt it for validity. Concrete examples of the procedure are given on pp. 897-899.

2 Semantical tableaux

2.1 General description  The basic notion is that of a column. Finite sets of columns make configurations. Configurations are patterned into trees that are tableaux.

A column can be viewed as a finite number of levels, divided into one middle level, some upper levels, and some lower levels. On the middle level is a relative integer $i$, the index of the column. Each upper level (if any) is filled by a pair $\{k, l\}$ of indices. Each lower level (if any) is filled by a signed sentence. It gives sense to speak of the sequent associated with a column, to say that $\{k, l\}$ stays above $i$, or that $(\pm)A$ stays under $i$, in the given column. A configuration is a finite set of columns, where there is at most one column for each index, and a column of index 0. A configuration is a successor of another one if it comes from it via one of the rules that we shall describe in 2.2.

A tableau is a finite tree, whose nodes are configurations, such that if a configuration has in the tableau one successor relative to some application of a rule, then it has also the other ones, if any. [We must systematically investigate the possibilities of each rule that we retain.]

2.2 Building tableaux  We start with a given configuration, the input. Then we go on according to the following rules:

**Rules for indices**

$I_1$ Introduce $\{0, i\}$ above $i$ (in a given column)
$I_2$ Introduce $\{i, i\}$ above $i$
$I_3$ If $\{i, j\}$ stays above $k$, and $\{k, l\}$ above $m$, introduce a new index $x$, strictly positive, together with $\{i, l\}$ above $x$ and $\{x, j\}$ above $m$.
$I_4$ If $\{0, i\}$ stays above $j$, and $\{j, k\}$ above $l$, introduce $\{i, k\}$ above $l$.
$I_5$ If $\{i, j\}$ stays above $k$, introduce $\{i, -k\}$ above $-j$.
$I_6$ Introduce $-i$ if it is not yet in the configuration, and if $i$ stays in some column.

The result of a systematic iteration of these rules is a sort of "free" relevant structure, if we interpret $i^*$ by $-i$ and $R(i, j, k)$ by "$\{i, j\}$ stays above $k".$ $I_1$-$I_6$ restate postulates $P_1$-$P_5$. $P_6$ is automatically satisfied. In the construction, we give a priority to positive indices, but it is only a matter of convenience.

**Rule of reiteration**

If $\{0, i\}$ stays above $j$ in the configuration, and if $+A$ stays under $i$, enter $+A$ under $j$.

**Rules for logical operators**

For each logical symbol, we have $A_n$ admission rule and $A_n$ omission rule.
Admission rules

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Rule</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\wedge$</td>
<td>$+A \wedge B$ stays under $i$</td>
<td>Enter $+A$ and $+B$ under $i$</td>
</tr>
<tr>
<td>$\vee$</td>
<td>$+A \vee B$ stays under $i$</td>
<td>Enter $+A$ under $i$ or $+B$ under $i$ (2 successors)</td>
</tr>
<tr>
<td>$\to$</td>
<td>$+A \to B$ stays under $i$</td>
<td>Enter $+B$ under $k$ or $-A$ under $j$ (2 successors)</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$+\neg A$ stays under $i$</td>
<td>Enter $-A$ under $-i$</td>
</tr>
<tr>
<td>$\forall$</td>
<td>$+\forall xF[x]$ stays under $i$</td>
<td>Enter $F[i]$ under $i$ [1 successor for each $i$]</td>
</tr>
</tbody>
</table>

Omission rules

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Rule</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\wedge$</td>
<td>$-A \wedge B$ under $i$</td>
<td>Enter $-A$ under $i$ or $-B$ under $i$ (2 successors)</td>
</tr>
<tr>
<td>$\vee$</td>
<td>$-A \vee B$ under $i$</td>
<td>Enter $-A$ and $-B$ under $i$</td>
</tr>
<tr>
<td>$\to$</td>
<td>$-A \to B$ under $i$</td>
<td>Introduce new positive indices $x$, $y$, then enter: $+A$ under $x$, $-B$ under $y$, ${i, x}$ above $y$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$-\neg A$ under $i$</td>
<td>Enter $+A$ under $-i$</td>
</tr>
<tr>
<td>$\forall$</td>
<td>$-\forall xF[x]$ under $i$</td>
<td>Enter $-F[a]$ under $i$, $a$ is a parameter not occurring in the configuration [1 successor for each choice of $a$]</td>
</tr>
</tbody>
</table>

Note that each formula occurring in a successor of a configuration is a subformula of some formula in it and each configuration is in a clear manner included in each of its successors.

2.3 Closure of tableaux The notion of tableaux generated from a given input is then clear. A branch of a tableau is as usual a sequence of consecutive configurations beginning with the input. A column is closed provided that there is an $A$ such that $+A$ and $-A$ stay in the column. A configuration is closed if one of its columns is closed. A branch in a tableau is closed if one of its configurations is closed (and then all its successors). A tableau is closed if all its branches are closed.

With each sequent we can associate the following canonical configuration:
Proposition 1 A sequent is normally valid if and only if the preceding configuration is the input of a closed tableau.

It is convenient to generalize the situation a bit. Let \( \mathcal{C} \) be a configuration. A countermodel of \( \mathcal{C} \) is given by:

\[
\mathfrak{R} = \langle K, 0, R, * \rangle.
\]

For each index \( i \) in \( \mathcal{C} \), a member \( a_i \) of \( K \) is such that:

- \( a_0 = 0 \)
- If \( i \) and \( -i \) are in \( \mathcal{C} \), then \( a_{-i} = a_i^* \)
- If \( i, j \) stays above \( k \) in \( \mathcal{C} \), then \( R(a_i, a_j, a_k) \)
- \( \langle \mathfrak{R}, a_i \rangle \) is a countermodel of the column at \( i \) in \( \mathcal{C} \).

Notice that necessarily \( 0 = 0^* \) in \( K \). So it is natural to state: A configuration is normally valid if it has no countermodel. Then a sequent is normally valid if and only if its canonical configuration is normally valid.

Remark: We could have defined validity for configurations by working with integers \( \neq 0 \), and replacing everywhere 0 by 1 (so that we do not force 0 and 0* to be equal).

Proposition 2 A configuration is normally valid if and only if it is the input of a closed tableau.

3 The "If"-side (soundness)

Lemma 1 If a configuration has a countermodel, then for each rule of Section 2, at least one of its successors has a countermodel.

Since no closed configuration can have a countermodel, half of Proposition 2 falls at once from Lemma 1.

Proof: Let us start with a configuration \( \mathcal{C} \) and a countermodel of it, based on a r.m.s. \( \mathfrak{R} = \langle K, 0, R, * \rangle \). We have to examine one after another each possibility of creating a successor. All verifications are routine:

Rules for indices \( I_1-I_6 \) correspond to postulates \( P_1-P_6 \) of [2]. Taking the worst case, suppose that we have just entered \( x \) by \( I_3 \), provided that \( \{i, j\} \) is above \( k \) and \( \{k, l\} \) above \( m \) in \( \mathcal{C} \). Then we know that we have in \( K \)

\[
R(a_i, a_j, a_k) \text{ and } R(a_k, a_l, a_m).
\]

So by \( P_3 \) there must be an \( a \) in \( K \) such that:

\[
R(a_i, a_j, a) \text{ and } R(a, a_j, a_m).
\]

It suffices to put \( a_x = a \). For \( I_6 \), we have just to put: \( a_{-j} = a_j^* \).
Reiteration Use monotony of forcing:
\[ \alpha \vdash A \text{ and } R(0, \alpha, \beta) \Rightarrow \beta \vdash A \text{ for every } A. \]

Admission rules +A \land B \text{ being under } i, \text{ we have just to enter } +A \text{ and } +B.

We know that \( \alpha_i \vdash A \land B \), hence \( \alpha_i \vdash A \) and \( \alpha_i \vdash B \). So the previous countermodel is yet good.

+A \lor B \text{ being under } i, \alpha_i \vdash A \lor B, \text{ hence } \alpha_i \vdash A \text{ or } \alpha_i \vdash B. \text{ If } \alpha_i \vdash A, \text{ choose to enter } +A. \text{ If not, choose to enter } +B. \text{ In all cases, the old countermodel works.}

+ A \rightarrow B \text{ stays under } i, \text{ and } \{i, j\} \text{ above } k. \text{ If } \alpha_j \vdash A, \text{ since } R(\alpha_i, \alpha_j, \alpha_k) \text{ then } \alpha_k \vdash B: \text{ enter } +B \text{ under } k. \text{ If } \alpha_j \not\vdash A, \text{ choose to enter } -A \text{ under } j. \text{ In both cases, the old model still works.}

We have just entered -A under -i, provided that +A stayed under i. Since we know that \( \alpha_i \vdash -A \), \( \alpha_i \not\vdash A \). But we have \( \alpha_i^* = \alpha_{-i} \).

We have just entered +F[t] under i, provided that +\forall x F[x] was there. Since \( \alpha_k \vdash \forall x F[x], \alpha_i \vdash F[t] \) where \( t \) is the value in \( D \) of \( t \), which is the same as \( \alpha_i \vdash F[t] \).

Omission rules Let us consider only the worst cases:

- A \rightarrow B \text{ stays under } i. \text{ Now we enter a new } j \text{ and a new } k \text{ together with } +A \text{ under } j, \text{ } -B \text{ under } k \text{ and } \{i, j\} \text{ above } k. \text{ Since } \alpha_i \vdash A \rightarrow B, \text{ there must be in } K a \text{ and } \beta \text{ which satisfy:}

\[ a \vdash A, \beta \not\vdash B \text{ and } R(\alpha_i, a, \beta) \]

and which provide convenient values for \( \alpha_j \) and \( \alpha_k \).

- \forall x F[x] \text{ stays under } i \text{ and we enter } -F[a] \text{ with a new } a. \text{ Since } \alpha_i \vdash \forall x F[x], \text{ there is some } \overline{a} \text{ in } D \text{ such that:}

\[ a \vdash -F[\overline{a}]. \]

a being a foreigner to the previous configuration, we are free to slightly modify \( \mathfrak{A} \), so that \( a \) can be interpreted by \( \overline{a} \), without changing the rest. We have yet a countermodel of the old configuration, but it now works for the new one.

So Lemma 1 is proved.

4 The "only-if"-side (completeness)

Lemma 2 If a configuration is not the input of some closed tableau, then it has a countermodel.

Proof: Let \( C_0 \) be the given configuration. Starting with \( T_0 = C_0 \), we build an increasing sequence of tableaux:

\[ T_0 \subset T_1 \subset \ldots \subset T_n \subset \ldots \]

such that each operation that could be performed on a terminal configuration of \( T_n \) is actually accomplished when building \( T_{n+1} \) [it is easy, though
boring, to make the procedure deterministic]. Of course, the "sweeping process" must ensure that if \(+VF[x]\) stays under \(i\) in some node of \(J_n\), then we successively enter all \(+F[i]\) when running through the \(J_n\)'s along any branch which passes by this node. Under the hypothesis of Lemma 2, no \(J_n\) is closed. So, applying König's lemma, we may find a way along the \(J_n\)'s which runs along a non-closed branch of each \(J_n\). Let \(C_n\) be the terminal configuration of this branch in \(J_n\). We get an increasing sequence:

\[
C_0 \subseteq C_1 \subseteq \ldots \subseteq C_n \subseteq \ldots
\]

Its union \(C = \bigcup_n C_n\) is a possibly infinite "configuration". We shall see that it defines a countermodel of \(C_0\).

Let \(K\) be the set of indices occurring in \(C\). Due to \(I_\theta\) and the "sweeping process", \(i \in K \Rightarrow -i \in K\). So we can define:

\[
i* = -i
\]

\[
R(i, j, k) \iff \{i, j\} \text{ stays above } k \text{ in } C.
\]

Rules for indices yield the validity of \(\text{P}_1-\text{P}_\theta\). So we have a (normal) r.m.s.

Take the set \(D\) of closed terms as domain, each individual or functional symbol having its canonical interpretation in the free algebra of terms. For each atomic sentence \(A\) and \(i \in K\), put:

\[
i \Vdash A \iff +A \text{ stays under } i \text{ in } C.
\]

Reiteration gives the monotony condition:

\[
i \Vdash A \text{ and } R(0, i, j) \Rightarrow j \Vdash A.
\]

To be done we just need the following:

**Lemma 3** For each \(i \in K\), let \(C_i\) be the \(i\)-column of \(C\):

- If \(+A \in C_i\), then \(i \Vdash A\)
- If \(-A \in C_i\), then \(i \not\Vdash A\).

A routine induction over \(A\).

\(A\) is atomic: \(+A \in C_i \Rightarrow i \Vdash A\) by definition. If \(-A \in C_i\), then \(+A \notin C_i\) since no \(C_n\) is closed. Hence \(i \not\Vdash A\).

\(A = \overline{B}\):

- If \(+A \in C_i\), then \(-B \in C_{i^*}. -i = i^* \Vdash B, \text{ so } i \Vdash \overline{B}\)
- If \(-A \in C_i\), then \(+B \in C_{i^*}. i^* \Vdash B, \text{ hence } i \not\Vdash \overline{B}\)

\(A = B \rightarrow C\):

- \(+A \in C_i\): Let any \(j\) such that \(j \Vdash -B\), and any \(k\) such that \(R(i, j, k)\), i.e., \(\{i, j\}\) above \(k\) in \(C_k\). It cannot be the case that \(-B \in C_j\), since we should have \(j \Vdash B\). So \(+C \in C_k\) and \(k \Vdash C\). Thus \(i \Vdash B \rightarrow C\).

- \(-A \in C_i\): then there are \(j\) and \(k\) such that \(+B \in C_j\), \(-C \in C_k\) and \(\{i, j\}\) stays above \(k\) in \(C\).

So \(j \Vdash B, k \Vdash C\) and \(R(i, j, k)\). Thus \(i \not\Vdash B \rightarrow C\).
A = B \land C, A = B \lor C are not problematic.

A = \forall x F[x]:

If +A \in C_i, we have ensured that for each t, +F[t] \in C_i.
So t \models F[t] for each t, and then t \not\models \forall x F[x]
If -A \in C_i, there is some a such that -F[a] \in C_i.
So t \not\models F[a], and of course t \models \forall x F[x].

5 Relation with the standard formulation of R.Q. Each theorem A of R.Q. is valid. Hence the configuration:

\[
\begin{array}{c|c}
0 & -A \\
\hline
\end{array}
\]

is the input of a closed tableau.

To prove the converse would be nothing but proving the completeness of R.Q. with respect to the above semantics. By [2] this result is known for the sentential part R of the calculus. So we do have a tableau-formulation adequate for R. For the time being, the author has not been able to achieve the same result for R.Q.

EXAMPLES

Proof of \((A \rightarrow \overline{B}) \rightarrow (B \rightarrow \overline{A})\)

\[
\begin{array}{c|c|c|c|c|c|c}
\hline
& 1, -3 & & & 1, 2 \\
\hline 1 & -3 & -2 & -1 & 0 & 1 & 2 & 3 \\
\hline
\hline
&A & & & & & & \\
\hline
-A & & & & & & & \\
\hline
+B & & & & & & & \\
\hline
\end{array}
\]
Proof of $\forall x(A \rightarrow B[x]) \rightarrow (A \rightarrow \forall x B[x])$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+\forall x(A \rightarrow B[x])$</td>
<td></td>
<td></td>
<td>+A</td>
<td></td>
</tr>
<tr>
<td>$-A \rightarrow \forall x B[x]$</td>
<td></td>
<td></td>
<td></td>
<td>-A</td>
</tr>
<tr>
<td>$+A \rightarrow B[a]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$-A$</td>
<td></td>
<td></td>
<td>-A</td>
<td></td>
</tr>
<tr>
<td>$+B[a]$</td>
<td></td>
<td>-B</td>
<td></td>
<td>+B</td>
</tr>
</tbody>
</table>

Proof of $(A \rightarrow \bar{A}) \rightarrow \bar{A}$

<table>
<thead>
<tr>
<th></th>
<th>-1, -1</th>
<th></th>
<th>-1, 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$+A$</td>
<td>-A</td>
<td></td>
<td>+A</td>
<td>-A</td>
</tr>
<tr>
<td>$-A$</td>
<td>-A</td>
<td></td>
<td>-A</td>
<td></td>
</tr>
<tr>
<td>$+\bar{A}$</td>
<td>+\bar{A}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Proof of $(A \rightarrow C) \land (B \rightarrow C) \rightarrow (A \lor B \rightarrow C)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+(A \rightarrow C) \land (B \rightarrow C)$</td>
<td></td>
<td></td>
<td>+A</td>
<td></td>
</tr>
<tr>
<td>$-A \lor B \rightarrow C$</td>
<td></td>
<td></td>
<td></td>
<td>-B</td>
</tr>
<tr>
<td>$+A \rightarrow C$</td>
<td></td>
<td></td>
<td>+A</td>
<td></td>
</tr>
<tr>
<td>$+B \rightarrow C$</td>
<td></td>
<td></td>
<td>+B</td>
<td></td>
</tr>
<tr>
<td>$-C$</td>
<td></td>
<td></td>
<td>-C</td>
<td></td>
</tr>
</tbody>
</table>

$\rightarrow$
REFERENCES


*Université Claude Bernard, Lyon I*
*Lyon, France*