An introduction into the relational calculus

The relational calculus is a calculus of mathematical relations. Pioneering work on the subject was done by Alfred Tarski as early as the forties of this century. In more recent times computing scientists have shown a renewed interest in the calculus, because of its potential adequacy for dealing with program semantics and even program development. The seminal paper by C.A.R. Hoare and He Jifeng [HH86], written in 1985, clearly marks the beginning of a period.

By a number of local circumstances we ourselves became involved with the relational calculus as well; being programmers, we came to ask ourselves how the calculus can be presented in a fashion so smooth and simple that it can be reconstructed with little effort at any moment, beit tomorrow, next week, or even a year from now. This text is an effort towards that goal. It presents the calculus in a number of small steps, each step comprising the introduction of one postulate and exploring the consequences of its incorporation.

The text was written with the uninitiated reader in mind. However, since we will present the relational calculus as a speciali-zation of the predicate calculus, familiarity

with the latter is needed. Our presentation is based on the notational conventions and the calculational mode laid down by Edsger W. Dijkstra and Carel S. Scholten in [DS90].

There are two papers from which we have benefitted greatly. The one is a technical report by Edsger W. Dijkstra [EWD1047] and the other a technical note by C. S. Scholten [CSS164]. In fact, we have tried to combine the best of both while adding our own cadenzas.

At earlier stages of our struggles with the relational calculus Jaap van der Woude has been very helpful, giving support and technical assistance. As always, the members of the Eindhoven Tuesday Afternoon Club have been a continuous and indispensable forum. There is one member in particular who initiated our involvement with the relational calculus and without whom we would never have written this paper. To him, to Carel Steven Scholten, we respectfully and gratefully dedicate this text, on the occasion of his becoming an honorary doctor.

0 A preamble on junctivity

When dealing with predicate transformers, i.e. functions from predicates to predicates, it may be quite useful to know their so-called junctivity properties. In this presentation of the relational calculus we will encounter some predicate transformers that have outstanding junctivity properties. Hence this preamble.

Usually one distinguishes two kinds of junctivity, viz. disjunctivity and conjunctivity.

The strongest form of disjunctivity is so-called universal disjunctivity. Predicate transformer f is called universally disjunctive if it distributes over arbitrary disjunctions, i.e. if for all possible ranges of dummy x,

$$[\, f.(\exists x :: x) \;\equiv\; (\exists x :: f.x)\,] \quad .$$

The best-known consequences of f being universally disjunctive are

- $[\, f.\mathit{false} \;\equiv\; \mathit{false}\,]$
- f is monotonic with respect to $\Rightarrow$, i.e. $[x \Rightarrow y] \;\Rightarrow\; [f.x \Rightarrow f.y]$.

The strongest form of conjunctivity is so-called universal conjunctivity. Predicate transformer g is called universally conjunctive if it distributes over arbitrary conjunctions, i.e. if for all possible ranges of dummy y,

$$[\ g.(\underline{A}y::y) \equiv (\underline{A}y::g.y)\ ] .$$

The best-known consequences of g being universally conjunctive are

- $[\ g.true \equiv true\ ]$
- g is monotonic with respect to $\Rightarrow$.

In what follows we shall freely use these facts.

* * *

Sometimes the universal junctivity of a predicate transformer comes for free, thanks to the following lemma.

<u>The junctivity lemma</u>

Let predicate transformers f and g be such that (for all x,y)

(*) $\quad [\ f.x \Rightarrow y\ ] \equiv [\ x \Rightarrow g.y\ ]$.

then f is universally disjunctive
and g is universally conjunctive .

<u>Proof</u> In order to prove f's universal disjunctivity we must prove, for arbitrary range of x,

(**) $\quad [\ f.(\underline{E}x::x) \equiv (\underline{E}x::f.x)\ ]$.

To that end we observe for any y ,

$[f.(\underline{E}x::x) \Rightarrow y]$

$=$ { (*) with $x := (\underline{E}x::x)$ }

$[(\underline{E}x::x) \Rightarrow g.y]$

$=$ { predicate calculus, i.e. distribution of $\Rightarrow g.y$ over $\underline{E}$ }

$[(\underline{A}x:: x \Rightarrow g.y)]$

$=$ { interchange of $[\,]$ and $\underline{A}$ }

$(\underline{A}x:: [x \Rightarrow g.y])$

$=$ { (*) }

$(\underline{A}x:: [f.x \Rightarrow y])$

$=$ { interchange of $\underline{A}$ and $[\,]$ }

$[(\underline{A}x:: f.x \Rightarrow y)]$

$=$ { predicate calculus }

$[(\underline{E}x:: f.x) \Rightarrow y]$,

and since the equivalence between the first and the last line holds for any y, (**) follows. Note that in the above proof the internal structure of the expression $g.y$ is completely irrelevant. That expression could equally well be something outlandish like $\neg(\sim z \,;\, \neg y)$, provided it is a predicate.

The — very similar — proof of g's universal conjunctivity is left to the reader.

(<u>End</u> of Proof.)

1 The transposition

In this presentation of the relational calculus relations are considered to be predicates, and the relational calculus will emerge from the predicate calculus by extending the latter with two new operators and a new constant.

The first operator to be added is the so-called transposition. It is a unary operator to be denoted by a prefix $\sim$ – pronounced "tilde" – and it is given the same binding power as $\neg$. By postulate, the transposition satisfies

(0) $[\sim x \Rightarrow y] \equiv [x \Rightarrow \sim y]$ (for all x, y).

From (0) and the junctivity lemma with $f, g := \sim, \sim$ we conclude

$\sim$ is universally disjunctive, and
$\sim$ is universally conjunctive,

and hence

(1) $[\sim false \equiv false]$, $[\sim true \equiv true]$,
$\sim$ is monotonic with respect to $\Rightarrow$.

After thus having settled $\sim$'s junctivity properties, we now investigate how it "behaves" in combination with negation, the square brackets, and itself.

As for the latter we will show that $\sim$ is an involution, i.e.

(2) $[\sim\sim x \equiv x]$ (for all x)

<u>Proof</u> We observe for any x

$$
\begin{array}{ll}
 & [\sim\sim x \Rightarrow x] \\
= & \quad \{(0), \text{ see Remark below}\} \\
 & [\sim x \Rightarrow \sim x] \\
= & \quad \{(0)\} \\
 & [x \Rightarrow \sim\sim x]
\end{array}
$$

Since the middle line equivales true, so do the outer lines, which establishes (2).

(<u>End</u> of Proof.)

<u>Remark</u> In the hints of the above little calculation we deliberately left out the particular instantiations of (0) because including them would not agree with our view of (0): formula (0) expresses that we can freely move the symbol $\sim$ back and forth between the antecedent and the consequent of an implication between square brackets. It is by means of this simple "symbol dynamics" that we remember and use (0). The situation is very similar to the way most people deal with e.g. the rules of de Morgan.
(<u>End</u> of Remark.)

Next we show that transposition and negation distribute over each other, i.e.

(3) $[\neg\sim x \equiv \sim\neg x]$ (for all x) .

<u>Proof</u> The proof is by mutual implication:

$[\neg\sim x \Rightarrow \sim\neg x]$
= { shunting the antecedent to the consequent }
$[\mathit{true} \Rightarrow \sim x \vee \sim\neg x]$,

and the validity of the latter follows from

$\sim x \vee \sim\neg x$
= { $\sim$ over $\vee$: $\sim$ is universally disjunctive }
$\sim(x \vee \neg x)$
= { predicate calculus }
$\sim \mathit{true}$
= { (1) }
true .

The -very similar- proof of the implication in the other direction is left to the reader.

(<u>End</u> of Proof.)

Because the transposition is universally junctive and distributes over the negation, we have as a corollary :

(4) Transposition distributes over all logical operators including the two logical quantifiers .

Finally, from (0) with $x, y := true, x$ together with $[\sim true \equiv true]$, we conclude

(5) $\quad [x] \equiv [\sim x] \quad$;

i.e. we can freely transpose or "de-transpose" an expression between square brackets.
This rule is useful for calculational practice, especially in combination with (4).

This completes our introduction of the transposition.

2 The composition and the transposition

The second - and last! - operator to be added to the predicate calculus is the so-called composition. It is a binary operator to be denoted by an infix "$;$" - pronounced "semi" - and it is given a higher binding power than the other binary operators and a lower binding power than the unary operators.

Our first postulate about the composition is very important but not too interesting in its own right. It reads

(0) the composition is associative, i.e.

$$[\, x\,;(y\,;z) \equiv (x\,;y)\,;z \,] \quad .$$

The second postulate is, in fact, a pair of postulates linking composition and transposition:

(1a) the "left-exchange"

$$[\, x\,;y \Rightarrow z \,] \equiv [\, \neg z\,; \sim y \Rightarrow \neg x \,]$$

(1b) the "right-exchange"

$$[\, x\,;y \Rightarrow z \,] \equiv [\, \sim x\,; \neg z \Rightarrow \neg y \,] \quad .$$

At first, these rules look horrifying but they no longer are once we become aware of their "symbol dynamics". First observe that all antecedents are compositions of two operands. In the left-exchange the one side of the equivalence is transformed into the other by exchanging the left operand and the consequent while negating both – i.e. taking their contrapositive – and transposing the operand that remains in place. Note that this recipe works in both directions since removing a negation or transposition is the same as adding one – both are involutions –. The right-exchange has the same dynamics, except that this time the right operand is exchanged.

The above exchange rules can be combined – about which more later – into Jaap van der Woude's rule:

(2) the "middle-exchange"

$$[x ; u ; y \Rightarrow v]$$
$$\equiv \quad [\sim x ; \neg v ; \sim y \Rightarrow \neg u] \quad .$$

<u>Proof</u>

$[x ; u ; y \Rightarrow v]$

= { (0), i.e. ; is associative }

$[(x ; u) ; y \Rightarrow v]$

= { left-exchange }

$[\neg v ; \sim y \Rightarrow \neg (x ; u)]$

= { contrapositive }

$[\; x ; u \;\Rightarrow\; \neg(\neg v ; \sim y) \;]$

= $\{$right-exchange$\}$

$[\; \sim x ; (\neg v ; \sim y) \;\Rightarrow\; \neg u \;]$

= $\{(0)\}$

$[\; \sim x ; \neg v ; \sim y \;\Rightarrow\; \neg u \;]$.

(<u>End</u> of Proof.)

<u>Remark</u> The exchange rules may bring about all sorts of functional compositions of $\neg$ and $\sim$. In our calculations we will not hesitate to simplify these compositions in one go along with other manipulations, even without saying so explicitly. Thus, the negation of $\sim\neg x$ simply is $\sim x$.
(<u>End</u> of Remark.)

* * *

Next we investigate whether $\sim$ has any distributivity properties with respect to composition. To that end we observe for any x, y, z

$[\; \sim(x ; y) \;\Rightarrow\; z \;]$

= $\{$ move $\sim$ to consequent (see (1.0)) $\}$

$[\; x ; y \;\Rightarrow\; \sim z \;]$

= $\{$ left-exchange $\}$

$[\; \neg\sim z ; \sim y \;\Rightarrow\; \neg x \;]$

= $\{$ right-exchange $\}$

$[\; \neg z ; x \;\Rightarrow\; \neg\sim y \;]$

= $\{$ left-exchange $\}$

$[\; \sim y ; \sim x \;\Rightarrow\; z \;]$

and since the equivalence between the first and the last line holds for any z , we conclude

(3) the "$\sim$ over $;$"-rule

$$[\sim(x ; y) \equiv \sim y ; \sim x] .$$

* * *

Next we show that composition is universally disjunctive in both operands We do so by exploiting the junctivity lemma dealt with in the preamble. For arbitrary z, we define predicate transformer f as follows :

$$[f.x \equiv z ; x] \qquad \text{(for all } x) .$$

By the junctivity lemma, we can prove f's universally disjunctivity - and. hence, the composition's universal disjunctivity in its second operand - provided we manage to establish

$$[f.x \Rightarrow y] \equiv [x \Rightarrow \text{something}] .$$

This turns out to be very simple :

$[f.x \Rightarrow y]$
$=$ { definition of f }
$[z ; x \Rightarrow y]$
$=$ { right- exchange in order to isolate x }
$[\sim z ; \neg y \Rightarrow \neg x]$
$=$ { contrapositive }
$[x \Rightarrow \neg(\sim z ; \neg y)]$
$=$ { }
$[x \Rightarrow \text{something}]$.

The -very similar- proof of the composition's universal disjunctivity in the first operand is left to the reader. Thus we have established

(4) the composition is universally disjunctive in both operands .

As a result, we also have

(5) $[\text{false} ; x \equiv \text{false}]$. $[x ; \text{false} \equiv \text{false}]$,

and

composition is monotonic with respect to $\Rightarrow$ in both operands .

* * *

Finally, we wish to mention that composition and negation do not relate nicely and that -hence- there are no nice conjunctivity properties of the composition.

3 The identity of composition

We would like composition to have an identity element. Therefore, we postulate the existence of a constant predicate J satisfying

(0) $[x ; J \equiv x]$ (for all x),

which conveys that J is a right-identity element.

Remark We cannot hope to prove the existence of such a J from our previous postulates, because – as Rob Hoogerwoord observed – all the latter are satisfied if the composition is implemented by $[x ; y \equiv false]$ for all x, y : in this case (0) reduces to the highly improbable $[false \equiv x]$ for all x.
(End of Remark.)

From (0) we derive that composition has $\sim J$ as a left-identity element:

(1) $[\sim J ; x \equiv x]$ (for all x)

Proof We observe for any x,

$\sim J ; x$
$=$ $\{ [\sim\sim x \equiv x]$, see (1.2) $\}$
$\sim J ; \sim\sim x$
$=$ $\{ \sim$ over $;$, see (2.3) $\}$

$$\begin{array}{ll}
 & \sim(\sim x \,;\, J) \\
= & \quad \{ (0) \text{ with } x := \sim x \} \\
 & \sim(\sim x) \\
= & \quad \{ \sim\sim \} \\
 & x
\end{array}$$

(<u>End</u> of Proof.)

If existent, left- and right-identities, are equal, so we have

(2) $[\sim J \equiv J]$,

which follows from

$$\begin{array}{ll}
 & \sim J \\
= & \quad \{ (0) \text{ with } x := \sim J \} \\
 & \sim J \,;\, J \\
= & \quad \{ (1) \text{ with } x := J \} \\
 & J \quad .
\end{array}$$

In summary we conclude from (0), (1), and (2)

(3) $[x \,;\, J \equiv x]$ and $[J \,;\, x \equiv x]$,

i.e. J is the (two-sided) identity element of composition.

The one and only theorem of this section that will be appealed to later on is

(4) $[x \Rightarrow x \,;\, true]$, and its dual
$[x \Rightarrow true \,;\, x]$.

Proof We prove $[x \Rightarrow x;\text{true}]$

$x;\text{true}$
$\Leftarrow$ { ; is monotonic, using $[\text{true} \Leftarrow J]$ }
$x;J$
$=$ {identity of ; }
x .

(End of Proof.)

With the introduction of J an overwhelming number of new theorems waits to be formulated or invented. In this introductory text we will, however, hardly touch on any of these. Presumably, the enormous growth in "manipulative potential" is best explained by the shape of formulae (3) and (4): they are the only rules, so far, for introducing (and removing) compositions.

* * *

The relational calculus can, of course, be developed in various ways. More traditional presentations, for instance, postulate at one fell swoop

- ; is associative
- J is the two-sided identity element of ;
- $[J \equiv \sim J]$
- the middle-exchange rule,

and this indeed suffices to prove all the results that we have established so far. The problem with this set of postulates, however, is that hardly anything useful can be derived from a proper subset of them. It is only when the whole bunch is taken into account that the calculus opens up like Pandora's Box. It is precisely this lack of disentangledness that has prompted us to diverge from such a more traditional presentation.

4 A little theory of left- and right-conditions

In this section we offer no new postulates but instead deal with two special kinds of predicates, to be called "left-conditions" and "right-conditions". We do so because in every treatment of the relational calculus these "conditions" pop up every so often and because an elementary mastery of their algebraic properties may have a beneficial effect on our relational calculations as a whole.

By definition,

(0a) p is a left-condition $\equiv$ $[p ; true \equiv p]$

(0b) q is a right-condition $\equiv$ $[true ; q \equiv q]$.

Because for all x, we have — see (3.4) —

$$[x ; true \Leftarrow x] \quad \text{and} \quad [true ; x \Leftarrow x] ,$$

we may conclude

(1a) p is a left-condition $\equiv$ $[p ; true \Rightarrow p]$

(1b) q is a right-condition $\equiv$ $[true ; q \Rightarrow q]$.

Note that formulae (1) are formally weaker characterizations of conditions than formulae (0).

<u>Remark</u> It is in general quite useful to be aware of the various equivalent forms of a formula, in particular if some of these

forms are formally weaker than others. For proving the validity of a formula, the weak version is usually to be preferred, whereas in exploiting the validity the strong version is preferable.
(End of Remark.)

Because of (2) below, all theorems on left- and right-conditions come in pairs; in what follows we therefore focus on left-conditions mainly.

(2) p is a left-condition
  $\equiv$ $\sim p$ is a right-condition

Proof

  p is a left-condition
$=$ { (0a) }
  $[p ; \text{true} \equiv p]$
$=$ { $[x] \equiv [\sim x]$, see (1.5), and
   $\sim$ over $\equiv$, see (1.4) }
  $[\sim (p ; \text{true}) \equiv \sim p]$
$=$ { $\sim$ over $;$, see (2.3), and
   $[\sim \text{true} \equiv \text{true}]$ }
  $[\text{true} ; \sim p \equiv \sim p]$
$=$ { (0b) }
  $\sim p$ is a right-condition

(End of Proof.)

* * *

First we investigate how to form new left-conditions from existing ones. In fact, we can show

(3) $x : p$ is a left-condition
$\Leftarrow$ p is a left-condition , for all x,

and

(4) all logical expressions built from left-conditions only are left-conditions .

The proof of (3) is left to the reader. As for (4), existential quantification, negation, and the constant true suffice to build all logical expressions, so that (4) follows if we can demonstrate

(4a) true is a left-condition

(4b) $\neg p$ is a left-condition
$\equiv$ p is a left-condition

(4c) $(\underline{E}\, p :: p)$ is a left-condition
$\Leftarrow$ $(\underline{A}\, p ::$ p is a left-condition$)$.

The proof of (4a) is left to the reader.

<u>Proof of (4b)</u>

By (1a), the result follows from

$[\neg p : true \Rightarrow \neg p]$
$=$ { left-exchange, and $[\sim true \equiv true]$ }
$[p : true \Rightarrow p]$.

(<u>End</u> of Proof.)

<u>Proof of (4c)</u>

By (0a), the result follows from

$(\exists p :: p) ; true$
$=$ { ; is universally disjunctive }
$(\exists p :: p ; true)$
$=$ { from the antecedent of (4c): $[p ; true \equiv p]$ }
$(\exists p :: p)$.

(<u>End</u> of Proof.)

* * *

Among all theorems involving left- or right-conditions, the following one ranks very high in importance.

(5) for left-condition p,
$$[(p \wedge x) ; y \equiv p \wedge x ; y] \quad \text{(for all } x, y)$$.

Its dual is

(6) for right-condition q,
$$[x ; (y \wedge q) \equiv x ; y \wedge q] \quad \text{(for all } x, y)$$.

(It is the shape of these rules that has triggered the names "left-condition" and right-condition.)

<u>Proof of (5)</u>

The proof is by mutual implication. In

the one direction we use that p is a left-condition and in the other direction we use that $\neg p$ is a left-condition .

- $[(p \wedge x) ; y \;\Rightarrow\; p \wedge x ; y]$

  This part is established by two independent weakenings of the antecedent, viz.

  $$\begin{array}{ll} & (p \wedge x) ; y \\ \Rightarrow & \qquad \{ ; \text{ is monotonic} \} \\ & x ; y \end{array}$$

  and

  $$\begin{array}{ll} & (p \wedge x) ; y \\ \Rightarrow & \qquad \{ ; \text{ is monotonic} \} \\ & p ; true \\ = & \qquad \{ p \text{ is a left-condition} \} \\ & p \;. \end{array}$$

- $[(p \wedge x) ; y \;\Leftarrow\; p \wedge x ; y]$

  Here we follow a heuristical advice that we owe to Lex Bijlsma — see Remark below — and rewrite the demonstrandum into the equivalent

  $[\neg p \vee (p \wedge x) ; y \;\Leftarrow\; x ; y]$.

  This, now, follows from

  $$\begin{array}{ll} & \neg p \vee (p \wedge x) ; y \\ = & \qquad \{ \neg p \text{ is a left-condition, i.e. } \\ & \qquad\qquad [\neg p \equiv \neg p ; true] \; \} \\ & \neg p ; true \;\vee\; (p \wedge x) ; y \\ \Leftarrow & \qquad \{ ; \text{ is monotonic} \} \\ & \neg p ; y \;\vee\; (p \wedge x) ; y \end{array}$$

$=$ { $;$ over $\vee$ }
$(\neg p \vee (p \wedge x)) ; y$
$=$ { predicate calculus }
$(\neg p \vee x) ; y$
$\Leftarrow$ { $;$ is monotonic }
$x ; y$

(End of Proof of (5).)

Remark As the reader will have noticed, many of our proofs take the form of linear calculations. Whenever the demonstrandum is an implication or an equivalence, i.e. an expression with two sides, the question arises at which side to start calculating. A very useful heuristic is to start at the more complicated side. But what if both sides are equally complicated? Bijlsma's advice is to try to massage the entire demonstrandum so as to make one side more complicated than the other. How this can be done is a separate concern. We think that in the above proof we have shown a successful application of Bijlsma's Principle.
(End of Remark.)

Now we are ready to prove what may be called the main theorem on "conditions", which reads

(7) the "left/right - composition":

for left-condition p and
right-condition q ,

$$[\, p ; q \ \equiv\ p \wedge q \,]\ .$$

Together with $[x ; J \equiv x]$ and $[J ; x \equiv x]$, it is one of the few simple rules from the relational calculus that admit the introduction or elimination of the composition operator.

<u>Proof of (7)</u>

```
    $p ; q$
=       { predicate calculus }
    $(p \wedge true) ; q$
=       { p is a left-condition, hence (5)
          applies with $x, y := true, q$ }
    $p \wedge true ; q$
=       { q is a right-condition }
    $p \wedge q$   .
```

(<u>End</u> of Proof.)

And this concludes our treatment of "conditions".

5 The Cone Rules

All our postulates so far, viz.

- $[\sim x \Rightarrow y] \equiv [x \Rightarrow \sim y]$
- $;$ is associative
- $[x ; y \Rightarrow z] \equiv [\neg z ; \sim y \Rightarrow \neg x]$
  $[x ; y \Rightarrow z] \equiv [\sim x ; \neg z \Rightarrow \neg y]$
- $[x ; J \equiv x]$,

are satisfied if the triple $(\sim \mid ; \mid J)$ is implemented by the triple (identity transformer $\mid \wedge \mid$ true) . This possibility will now be excluded by our next and last postulate, that we owe to C.S. Scholten and has been dubbed "the Cone Rule" by Jaap van der Woude. It reads

(0) $[\text{true} ; \neg x ; \text{true}] \Leftarrow \neg [x]$ (for all x) .

It can be formally strengthened into

(1) $[\text{true} ; \neg x ; \text{true}] \equiv \neg [x]$,

because the implication

(2) $[\text{true} ; \neg x ; \text{true}] \Rightarrow \neg [x]$

follows from the previous postulates. The proof of (2) , which is by a case distinction between $[x] \equiv \text{true}$ and $[x] \equiv \text{false}$, is left to the reader

* * *

Edsger W. Dijkstra designed another rule with the effect of distinguishing the triples. It reads

(3a) $[x ; \text{true} \vee \text{true} ; y] \Rightarrow [x ; \text{true}] \vee [\text{true} ; y]$.

or -equivalently-

(3b) for all left-conditions p and right-conditions q,
$[p \vee q] \Rightarrow [p] \vee [q]$.

Alfred Tarski invented yet another rule with the same effect, viz.

(4) $[x ; \text{true}] \vee [\text{true} ; \neg x]$.

In the remaining part of this section we shall show that Scholten's (0), Dijkstra's (3), and Tarski's (4) are all equivalent. We do so by showing

$(0) \Leftarrow (4)$, $(4) \Leftarrow (3a)$, and $(3b) \Leftarrow (0)$

in this (irrelevant) order.

<u>Proof of $(0) \Leftarrow (4)$</u>

First we rewrite (4) into an equivalent

implicative shape, viz.

(*) $\quad [x ; \mathrm{true}] \Leftarrow \neg[\mathrm{true} ; \neg x]$

and prove $(0) \Leftarrow (*)$. We observe for any x

$$
\begin{array}{ll}
 & (0) \\
= & \quad \{ \text{definition of } (0) \} \\
 & [\mathrm{true} ; \neg x ; \mathrm{true}] \Leftarrow \neg[x] \\
= & \quad \{ ; \text{ is associative} \} \\
 & [(\mathrm{true} ; \neg x) ; \mathrm{true}] \Leftarrow \neg[x] \\
\Leftarrow & \quad \{ (*) \text{ with } x := (\mathrm{true} ; \neg x) \} \\
 & \neg[\mathrm{true} ; \neg(\mathrm{true} ; \neg x)] \Leftarrow \neg[x] \\
= & \quad \{ \text{contrapositive} \} \\
 & [\mathrm{true} ; \neg(\mathrm{true} ; \neg x)] \Rightarrow [x] \\
\Leftarrow & \quad \{ \text{monotonicity of } [\,] \} \\
 & [\mathrm{true} ; \neg(\mathrm{true} ; \neg x) \Rightarrow x] \\
= & \quad \{ \text{right-exchange, and } [\sim \mathrm{true} \equiv \mathrm{true}] \} \\
 & [\mathrm{true} ; \neg x \Rightarrow \mathrm{true} ; \neg x] \\
= & \quad \{ \text{predicate calculus} \} \\
 & \mathrm{true} \; .
\end{array}
$$

(<u>End</u> of Proof.)

<u>Proof of $(4) \Leftarrow (3a)$</u>

We observe for any x

$$
\begin{array}{ll}
 & (4) \\
= & \quad \{ \text{definition of } (4) \} \\
 & [x ; \mathrm{true}] \vee [\mathrm{true} ; \neg x] \\
\Leftarrow & \quad \{ (3a) \text{ with } y := \neg x \} \\
 & [x ; \mathrm{true} \vee \mathrm{true} ; \neg x] \\
\Leftarrow & \quad \{ [x ; \mathrm{true} \Leftarrow x] \text{ and } [\mathrm{true} ; \neg x \Leftarrow \neg x] , \\
 & \qquad \text{see } (3.4) \} \\
 & [x \vee \neg x]
\end{array}
$$

$=$ { predicate calculus }

  true .

(*End* of Proof.)

Proof of (3b) ⇐ (0)

For any left-condition p and right-condition q we observe

  $[p] \vee [q]$

$\Leftarrow$ { contrapositive of (0), i.e. $[x] \Leftarrow \neg[\mathit{true} ; \neg x ; \mathit{true}]$ }

  $\neg[\mathit{true} ; \neg p ; \mathit{true}] \vee \neg[\mathit{true} ; \neg q ; \mathit{true}]$

$=$ { $\neg p$ is a left-condition, i.e. $[\neg p ; \mathit{true} \equiv \neg p]$, and $\neg q$ is a right-condition, i.e. $[\mathit{true} ; \neg q \equiv \neg q]$ }

  $\neg[\mathit{true} ; \neg p] \vee \neg[\neg q ; \mathit{true}]$

$=$ { de Morgan }

  $\neg([\mathit{true} ; \neg p] \wedge [\neg q ; \mathit{true}])$

$=$ { [] over $\wedge$ }

  $\neg[\mathit{true} ; \neg p \wedge \neg q ; \mathit{true}]$

$=$ { $\mathit{true} ; \neg p$ is a left-condition because $\neg p$ is – see (4.3) – and $\neg q ; \mathit{true}$ is a right-condition because $\neg q$ is . Hence the left/right – composition applies – see (4.7) – }

  $\neg[\mathit{true} ; \neg p ; \neg q ; \mathit{true}]$

$=$ { $;$ is associative }

$\neg[\mathit{true} ; (\neg p ; \neg q) ; \mathit{true}]$

$=$ { left/right - composition on $\neg p ; \neg q$ }

$\neg[\mathit{true} ; (\neg p \wedge \neg q) ; \mathit{true}]$

$=$ { de Morgan }

$\neg[\mathit{true} ; \neg(p \vee q) ; \mathit{true}]$

$\Leftarrow$ { contrapositive of (2), i.e. $\neg[\mathit{true} ; \neg x ; \mathit{true}] \Leftarrow [x]$, with $x := p \vee q$ }

$[p \vee q]$

Remark The two strengthening steps in the above calculation could have been replaced with equality - preserving steps had we used the formally stronger (1) rather than (0) and (2) separately. This, however, would have obfuscated that the first strengthening is a genuine reference to the newly introduced postulate (0) or (1), whereas the second strengthening does not rely on that postulate at all. This confirms Lincoln A. Wallen's warning that the notion of "equivalence" should be handled with care: equivalences hide why the one side implies the other, and these reasons can radically differ for the two directions. (End of Remark.)

(End of Proof.)

And this concludes our introduction of the relational calculus.

5 Etudes

For the reader who has travelled this far we include a small set of exercises. We have not tried to sort them according to section or difficulty, simply because we do not really know how to do that.

From our own experience we know that many of these exercises can be discouragingly hard, even though all of them admit relatively short and technically simple solutions. Our advice to the reader who gets stuck is to stop trying, and reinvestigate the problem much more consciously than before, keeping one eye on the formula to be manipulated and the other on the manipulative possibilities offered by the calculus. During that process, application of some of the heuristical rules that we scattered through the text may help. The reader will then experience that the exercises, when carried out along these lines, are much more than just exercises in relational calculation: they are exercises in proof construction.

0. $[x \Rightarrow J] \wedge [y \Rightarrow J] \Rightarrow [x ; y \equiv x \wedge y]$

1. $[x \Rightarrow J] \Rightarrow [x \equiv \sim x]$

2. For p a left- or right-condition
$[p ; p \equiv p]$

3. $[\neg(x ; \text{true})] \equiv [\neg x]$

4. Alfred Tarski's rotation rule:
$[x ; y \Rightarrow \neg\sim z] \equiv [y ; z \Rightarrow \neg\sim x]$.

5. For p a left-condition
$[x ; (p \wedge y) \equiv (x \wedge \sim p) ; y]$

6. $[x \Rightarrow x ; \sim x ; x]$

7. For q a right-condition
$[q ; \text{true} ; \sim q \equiv q ; \sim q]$

8. $[J \equiv (\underline{A} x :: \neg(\sim x ; \neg x))]$

9. $[\neg(x ; \text{true} ; y)] \equiv [\neg x] \vee [\neg y]$
(this is yet another Cone Rule)

10. $[x \Rightarrow J] \Rightarrow [x \equiv (x ; \text{true} ; x) \wedge J]$

11. $[x ; y \wedge z \Rightarrow x ; (y \wedge \sim x ; z)]$

12. $[x \Rightarrow J] \;\Rightarrow\; [x;(y \wedge z) \equiv x;y \wedge z]$

13. For p and q left-conditions
$[p;q \equiv p] \;\vee\; [q]$

14. $[J] \;\vee\; [\neg J;\mathit{true}]$

6 References

[DS90] Predicate Calculus and Program Semantics, Edsger W. Dijkstra and Carel S. Scholten, Springer Verlag 1990

[EWD1047] A relational summary. Edsger W. Dijkstra. November 1990, Austin TX, USA

[HH86] The weakest prespecification, C.A.R. Hoare and He Jifeng, Fundamenta Informaticae 9: 51 - 84, 217 - 252, 1986

[CSS164] Introduction of transposition and composition, C.S. Scholten. January 1991, Beekbergen, The Netherlands.

Eindhoven,
8 March 1991

W.H.J. Feijen and A.J.M. van Gasteren,
Department of Mathematics and
Computing Science,
Eindhoven University of Technology,
P.O. Box 513,
5600 MB Eindhoven,
The Netherlands