Formal derivation of a Stable Marriage algorithm

A. Bijlsma*

January, 1991

1 Problem description

In this paper the well-known Stable Marriage Problem is considered once again. The name of this programming problem comes from the terms in which it was first described [2]:

> A certain community consists of n men and n women. Each person ranks those of the opposite sex in accordance with his or her preferences for a marriage partner. We seek a satisfactory way of marrying off all members of the community. (...) We call a set of marriages unstable (...) if under it there are a man and a woman who are not married to each other but prefer each other to their actual mates.
>
> QUESTION: For any pattern of preferences is it possible to find a stable set of marriages?

The purpose of the colourful anthropomorphic terminology seems to be to provide motivation for the design decisions. It was faithfully adhered to or even embellished upon by later authors on the subject. Among the concepts introduced in this fashion are *courtships* [2], *proposers held in suspense* [9], *jilted suitors* [6], *fiancés* (and even, though probably by mistake, *financés*) [1], *admirers* [4], *matchmakers* [3], and *equal rights for women* [12] as well as *chains of forced sacrifices of women* [7]. However, it has often been observed that in general such terminology encourages the programmer to make implicit assumptions and to refrain from exploring certain alternatives that do not fit easily into the metaphor. In this paper an alternative strategy is pursued in that a solution to the Stable Marriage Problem is derived by means of the formal manipulation of uninterpreted formulae, rigorously avoiding the usual imagery. The reader is invited to refrain from attaching romantic interpretations to the predicates introduced below.

Let us restate the problem in a more formal way. Given are two disjoint finite sets X and Y with the same number of elements. For convenience's sake, we establish the convention that dummies x, u range over X and dummies y, v range over Y. For every x there is given a linear order $\sqsupseteq_x$ of Y and for every y there is given a linear order $\sqsupseteq_y$ of X. We are asked to give variable f of type **array** X **of** Y a value that satisfies

$$(\forall y :: (\#x :: f.x = y) \leq 1)$$

*Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.

—i.e., f is injective—and

$$(\forall x, u :: x \sqsupseteq_{f.u} u \ \vee \ f.u \sqsupseteq_x f.x) \ .$$

2 First approximation

Our derivation starts with the replacement, in the postconditions, of a constant, namely X, by a variable: we introduce a variable A of type **set of** X and take as invariants[1]

$$\begin{array}{ll} P0: & (\forall y :: (\#x : A.x : f.x = y) \leq 1) \ , \\ P1: & (\forall x, u : A.x \wedge A.u : x \sqsupseteq_{f.u} u \ \vee \ f.u \sqsupseteq_x f.x) \ . \end{array}$$

These invariants are trivially initialized by $A := \emptyset$. An obvious strategy is to increase A by one element at a time until $A = X$. Thus our first approximation is

$$\begin{array}{l} |[\ \textbf{var} \ A\colon \textbf{set of} \ X\,; \\ \quad A := \emptyset \\ \quad \{\text{inv} \ P0..1; \ \text{bd} \ |X \setminus A|\} \\ ; \ \textbf{do} \ A \neq X \rightarrow \ |[\ \textbf{var} \ a\colon \ X\,; \\ \qquad\qquad\qquad\qquad S0 \ \{\neg A.a\} \\ \qquad\qquad\qquad ; \ \ A := A \cup \{\!|a|\!\} \\ \qquad\qquad\qquad]| \\ \quad \textbf{od} \\ ]| \ . \end{array}$$

3 Second approximation

It is now our task to find a solution for $S0$ as specified by the first approximation. Observing that $P0$ and $P1$ only mention the values of f on A leads us to suspect that $S0$ will involve giving a value to $f.a$. Therefore we investigate under what circumstances the assignment $f.a := b$ ensures the invariance of $P0..1$. Assuming $P0..1$ and $\neg A.a$, we derive[2]

$$\begin{array}{ll} & P0(A := A \cup \{\!|a|\!\})(f := f(a : b)) \\ \equiv & \quad \{\text{substitution}\} \\ & (\forall y :: (\#x : A.x \ \vee \ x = a : f(a : b).x = y) \leq 1) \\ \equiv & \quad \{\text{split off } x = a, \text{ using } \neg A.a\} \\ & (\forall y :: (\#x : A.x : f.x = y) + \lceil b = y \rceil \leq 1) \\ \equiv & \quad \{\text{split off } y = b\} \\ & (\forall y : y \neq b : (\#x : A.x : f.x = y) \leq 1) \ \wedge \ (\#x : A.x : f.x = b) = 0 \\ \equiv & \quad \{\text{first conjunct follows from } P0\} \\ & (\#x : A.x : f.x = b) = 0 \\ \equiv & \quad \{\text{property of } \#\} \\ & (\forall x : A.x : f.x \neq b) \ . \end{array}$$

[1] We allow ourselves to write $A.x$ as an abbreviation for $x \in A$.

[2] Here $f(a : b)$ is defined by $f(a : b).x = f.x$ for $x \neq a$ and $f(a : b).a = b$. Moreover, $\lceil \ldots \rceil$ denotes the conversion function from booleans to integers that is defined by $\lceil \mathit{true} \rceil = 1$ and $\lceil \mathit{false} \rceil = 0$.

We see that the assignment $f.a := b$ ensures the invariance of $P0$ provided b is chosen such that

$$Q0: \qquad (\forall x : A.x : f.x \neq b)$$

holds. Now we turn to $P1$. Under the same assumptions as before we have

$$\begin{array}{ll}
& P1(A := A \cup \{\!|a|\!\})(f := f(a:b)) \\
\equiv & \quad \{\text{substitution}\} \\
& (\forall x, u : (A.x \ \vee \ x = a) \ \wedge \ (A.u \ \vee \ u = a) : x \sqsupseteq_{f(a:b).u} u \ \vee \ f(a:b).u \sqsupseteq_x f(a:b).x) \\
\equiv & \quad \{\text{split off } x = a \text{ and split off } u = a, \text{ using } \neg A.a\} \\
& (\forall x, u : A.x \wedge A.u : x \sqsupseteq_{f.u} u \ \vee \ f.u \sqsupseteq_x f.x) \\
& \wedge \ (\forall u : A.u : a \sqsupseteq_{f.u} u \ \vee \ f.u \sqsupseteq_a b) \\
& \wedge \ (\forall x : A.x : x \sqsupseteq_b a \ \vee \ b \sqsupseteq_x f.x) \\
& \wedge \ (a \sqsupseteq_b a \ \vee \ b \sqsupseteq_a b) \\
\equiv & \quad \{\text{first conjunct is } P1; \\
& \quad \ \text{rename dummy } u := x \text{ in second conjunct;} \\
& \quad \ \text{fourth conjunct follows from reflexivity of } \sqsupseteq_b \text{ and } \sqsupseteq_a \\
& \quad \} \\
& (\forall x : A.x : (a \sqsupseteq_{f.x} x \ \vee \ f.x \sqsupseteq_a b) \ \wedge \ (x \sqsupseteq_b a \ \vee \ b \sqsupseteq_x f.x)) \ .
\end{array}$$

So we see that the assignment $f.a := b$ ensures the invariance of $P1$ provided a and b are chosen to satisfy

$$(\forall x : A.x : (a \sqsupseteq_{f.x} x \ \vee \ f.x \sqsupseteq_a b) \ \wedge \ (x \sqsupseteq_b a \ \vee \ b \sqsupseteq_x f.x)) \ . \tag{1}$$

However, this raises the question whether such a and b always exist. Unfortunately, the answer is negative. To see this, take $X = \{\!|0,1,2|\!\}$, $Y = \{\!|3,4,5|\!\}$ and let all orderings be the usual ordering of the integers. Consider a state where $A = \{\!|0,2|\!\}$, $f.0 = 4$, $f.2 = 5$. Then the condition $\neg A.a$ forces us to choose $a = 1$ and (1) becomes

$$(1 \geq 0 \vee 4 \geq b) \ \wedge \ (0 \geq 1 \vee b \geq 4) \ \wedge \ (1 \geq 2 \vee 5 \geq b) \ \wedge \ (2 \geq 1 \vee b \geq 5) \ ,$$

which is equivalent to

$$4 \leq b \leq 5 \ .$$

Clearly no value of b satisfying $Q0$ solves this equation.

It is therefore necessary to strengthen the repetition's invariant in order to make choosing a and b easier. Now the problem with this is that all four terms in (1) contain a or b or both. Since a and b are chosen afresh in each iteration step, incorporation of such a term into the invariant only helps if a and b are replaced therein by the dummies of suitable universal quantifications. Obviously, we get the simplest additional invariant if we do this for a term that contains only one of a and b, not both. There are two such terms; we let our choice be guided by the fact that the condition for a (viz., $\neg A.a$) is simpler than the one for b (viz., $(\forall x : A.x : f.x \neq b)$)—an asymmetry caused by the decision to represent the coupling between X and Y as a mapping. We therefore decide to strengthen the invariant of the repetition with

$$P2: \qquad (\forall x, u : A.x \wedge \neg A.u : u \sqsupseteq_{f.x} x) \ .$$

This does indeed simplify the choice of a and b, as (1) follows from $P2 \wedge Q1$, where

$$Q1: \qquad (\forall x : A.x : x \sqsupseteq_b a \ \vee \ b \sqsupseteq_x f.x) \ .$$

We have yet to check the invariance of $P2$. As $P2$ clearly holds when $A = \emptyset$, the initialization does not have to be adjusted. Moreover, assuming $P2$ we have

$$\begin{array}{ll}
& P2(A := A \cup \{a\})(f := f(a:b)) \\
\equiv & \quad \{\text{substitution}\} \\
& (\forall x, u : (A.x \ \vee \ x = a) \ \wedge \ \neg A.u \ \wedge \ u \neq a : u \sqsupseteq_{f(a:b).x} x) \\
\equiv & \quad \{\text{split off } x = a, \text{ using } \neg A.a\} \\
& (\forall x, u : A.x \ \wedge \ \neg A.u \ \wedge \ u \neq a : u \sqsupseteq_{f.x} x) \ \wedge \ (\forall u : \neg A.u \ \wedge \ u \neq a : u \sqsupseteq_b a) \\
\equiv & \quad \{\text{first conjunct follows from } P2\} \\
& (\forall u : \neg A.u \ \wedge \ u \neq a : u \sqsupseteq_b a) \\
\equiv & \quad \{\text{reflexivity, renaming}\} \\
& (\forall x : \neg A.x : x \sqsupseteq_b a) \ .
\end{array}$$

Hence the assignment $f.a := b$ ensures the invariance of $P2$ provided that a and b are chosen to satisfy

$$Q2: \qquad (\forall x : \neg A.x : x \sqsupseteq_b a) \ .$$

Hence the second approximation to the program is

$$\begin{array}{l}
|[\ \textbf{var} \ A\text{: } \textbf{set of} \ X\,; \\
\quad A := \emptyset \\
\quad \{\text{inv } P0..2\,;\ \text{bd } |X \setminus A|\} \\
;\ \ \textbf{do} \ A \neq X \rightarrow \ |[\ \textbf{var} \ a\text{: } X;\ b\text{: } Y\,; \\
\qquad\qquad\qquad\qquad S1\ \{\neg A.a \wedge Q0..2\} \\
\qquad\qquad\qquad ;\ \ f.a := b \\
\qquad\qquad\qquad ;\ \ A := A \cup \{a\} \\
\qquad\qquad\qquad]| \\
\quad \textbf{od} \\
]| \ .
\end{array}$$

4 Third approximation

For any b, it is not difficult to find a value of a that satisfies $Q1$ and $Q2$, as these both require a to be small in terms of $\sqsupseteq_b$. Indeed,

$$\begin{array}{ll}
& Q1..2 \\
\equiv & \quad \{\text{by definition}\} \\
& (\forall x : A.x : x \sqsupseteq_b a \ \vee \ b \sqsupseteq_x f.x) \ \wedge \ (\forall x : \neg A.x : x \sqsupseteq_b a) \\
\equiv & \quad \{\text{trading}\} \\
& (\forall x : A.x \ \wedge \ b \sqsubset_x f.x : x \sqsupseteq_b a) \ \wedge \ (\forall x : \neg A.x : x \sqsupseteq_b a) \\
\equiv & \quad \{\text{domain merge}\} \\
& (\forall x : (A.x \ \wedge \ b \sqsubset_x f.x) \ \vee \ \neg A.x : x \sqsupseteq_b a) \\
\equiv & \quad \{\text{complement rule}\} \\
& (\forall x : b \sqsubset_x f.x \ \vee \ \neg A.x : x \sqsupseteq_b a) \ .
\end{array}$$

Having now shown that

$$Q1 \wedge Q2 \;\equiv\; (\forall x : b \sqsubset_x f.x \;\vee\; \neg A.x : x \sqsupseteq_b a) \;\;, \tag{2}$$

we find that $Q1..2$ may be initialized by $a := (\sqcap_b\, x :: x)$, where $\sqcap_b$ is the minimum quantifier corresponding to $\sqsupseteq_b$. Of course, there is no reason to suppose that this a satisfies $\neg A.a$. The search for a suitable a will be organized as a repetition with invariant $Q0..2$ and guard $A.a$. Our program now looks like this:

$|[$ **var** A: **set of** X;
  $A := \emptyset$
  {inv $P0..2$; bd $|X \setminus A|$}
; **do** $A \neq X \rightarrow$ $|[$ **var** a: X; b: Y;
                $S2$ {$Q0$}
              ; $a := (\sqcap_b\, x :: x)$
                {inv $P0..2 \wedge Q0..2$; bd $(\#x :: x \sqsupseteq_b a)$}
              ; **do** $A.a \rightarrow S3$ **od**
              ; $f.a := b$
              ; $A := A \cup \{a\}$
              $]|$
  od
$]|$.

Note that the bound function represents a decision to inspect candidates for a in increasing order with respect to $\sqsupseteq_b$.

5 A simple program

Let us investigate solutions for $S3$ of the form $a := a'$. Assuming $P0..2 \wedge Q0..2 \wedge A.a$ we have

$$\begin{array}{ll}
 & Q1..2(a := a') \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x : A.x : x \sqsupseteq_b a' \;\vee\; b \sqsupseteq_x f.x) \;\wedge\; (\forall x : \neg A.x : x \sqsupseteq_b a') \\
\equiv & \quad \{\text{split off } x = a, \text{ using } A.a\} \\
 & a \sqsupseteq_b a' \;\vee\; b \sqsupseteq_a f.a \\
 & \wedge\; (\forall x : A.x \;\wedge\; x \neq a : x \sqsupseteq_b a' \;\vee\; b \sqsupseteq_x f.x) \\
 & \wedge\; (\forall x : \neg A.x : x \sqsupseteq_b a') \\
\Leftarrow & \quad \{\text{heading for } a' \sqsupseteq_b a, \text{ in order to decrease the bound function}\} \\
 & b \sqsupseteq_a f.a \\
 & \wedge\; (\forall x : A.x \;\wedge\; x \neq a : x \sqsupseteq_b a' \;\vee\; b \sqsupseteq_x f.x) \\
 & \wedge\; (\forall x : \neg A.x : x \sqsupseteq_b a') \\
\Leftarrow & \quad \{Q1..2 \text{ implies last two terms with } a \text{ instead of } a'\} \\
 & b \sqsupseteq_a f.a \\
 & \wedge\; (\forall x : A.x \;\wedge\; x \neq a : x \sqsupseteq_b a \Rightarrow x \sqsupseteq_b a') \\
 & \wedge\; (\forall x : \neg A.x : x \sqsupseteq_b a \Rightarrow x \sqsupseteq_b a') \\
\equiv & \quad \{\text{merge domains of last two conjuncts, using } A.a\} \\
 & b \sqsupseteq_a f.a \;\;\wedge\;\; (\forall x : x \neq a : x \sqsupseteq_b a \Rightarrow x \sqsupseteq_b a')
\end{array}$$

$$
\begin{array}{ll}
\equiv & \{\text{trading}\} \\
& b \sqsupseteq_a f.a \ \wedge \ (\forall x :: x \sqsupset_b a \Rightarrow x \sqsupseteq_b a') \\
\Leftarrow & \{(\exists x :: x \sqsupset_b a), \text{ see proof below}\} \\
& b \sqsupseteq_a f.a \ \wedge \ a' = (\sqcap_b x : x \sqsupset_b a : x) \ .
\end{array}
$$

The existential quantification in the last hint of this derivation holds because

$$
\begin{array}{ll}
& (\exists x :: x \sqsupset_b a) \\
\equiv & \{\text{definition of } \sqsupset_b, \text{ trading}\} \\
& (\exists x : x \neq a : x \sqsupseteq_b a) \\
\Leftarrow & \{A.a\} \\
& (\exists x : \neg A.x : x \sqsupseteq_b a) \\
\equiv & \{Q2\} \\
& (\exists x :: \neg A.x) \\
\equiv & \{\} \\
& A \neq X \ .
\end{array}
$$

As the invariance of $P0..2 \wedge Q0$ is trivial, it has now been established that the assignment $a := (\sqcap_b x : x \sqsupset_b a : x)$ satisfies the requirements for $S3$, provided $b \sqsupseteq_a f.a$ holds. What if it does not? In that case, the simplest[3] plausible way to establish the required inequality would be a swap $f.a, b := b, f.a$. Such a swap trivially preserves $A.a$; we are obliged to check, however, that it also leaves the invariants intact.

$$
\begin{array}{ll}
& P0(f := f(a:b)) \\
\equiv & \{\text{substitution}\} \\
& (\forall y :: (\#x : A.x : f(a:b).x = y) \leq 1) \\
\equiv & \{\text{split off } x = a, \text{ using } A.a\} \\
& (\forall y :: (\#x : A.x \ \wedge \ x \neq a : f.x = y) + \lceil b = y \rceil \leq 1) \\
\equiv & \{\text{split off } y = b\} \\
& (\forall y : y \neq b : (\#x : A.x \ \wedge \ x \neq a : f.x = y) \leq 1) \ \wedge \ (\#x : A.x \ \wedge \ x \neq a : f.x = b) = 0 \\
\equiv & \{\text{first conjunct follows from } P0, \text{ second from } Q0\} \\
& true \ ;
\end{array}
$$

$$
\begin{array}{ll}
& P1(f := f(a:b)) \\
\equiv & \{\text{substitution}\} \\
& (\forall x, u : A.x \wedge A.u : x \sqsupseteq_{f(a:b).u} u \ \vee \ f(a:b).u \sqsupseteq_x f(a:b).x) \\
\equiv & \{\text{split off } x = a \text{ and } u = a, \text{ using } A.a\} \\
& (\forall x : A.x \ \wedge \ x \neq a \ \wedge \ A.u \ \wedge \ u \neq a : x \sqsupseteq_{f.u} u \ \vee \ f.u \sqsupseteq_x f.x) \\
& \wedge \ (\forall u : A.u \ \wedge \ u \neq a : a \sqsupseteq_{f.u} u \ \vee \ f.u \sqsupseteq_a b) \\
& \wedge \ (\forall x : A.x \ \wedge \ x \neq a : x \sqsupseteq_b a \ \vee \ b \sqsupseteq_x f.x) \\
& \wedge \ (a \sqsupseteq_b a \ \vee \ b \sqsupseteq_a b) \\
\equiv & \{\text{first conjunct follows from } P1; \\
& \ \text{rename dummy } u := x \text{ in second conjunct}; \\
& \ \text{fourth conjunct follows from reflexivity of } \sqsupseteq_b \text{ and } \sqsupseteq_a \\
& \}
\end{array}
$$

[3] A single assignment $f.a := b$ or $b := f.a$ is ruled out by $Q0$.

$$
\begin{array}{ll}
 & (\forall x : A.x \ \wedge \ x \neq a : (x \sqsupseteq_b a \ \vee \ b \sqsupseteq_x f.x) \ \wedge \ (a \sqsupseteq_{f.x} x \ \vee \ f.x \sqsupseteq_a b)) \\
\equiv & \quad \{Q1\} \\
 & (\forall x : A.x \ \wedge \ x \neq a : a \sqsupseteq_{f.x} x \ \vee \ f.x \sqsupseteq_a b) \\
\Leftarrow & \quad \{\text{this follows from } P1 \wedge A.a \text{ with } f.a \text{ instead of } b\} \\
 & (\forall x :: f.x \sqsupseteq_a f.a \Rightarrow f.x \sqsupseteq_a b) \\
\Leftarrow & \quad \{\text{transitivity of } \sqsupseteq_a\} \\
 & f.a \sqsupseteq_a b \ ;
\end{array}
$$

$$
\begin{array}{ll}
 & P2(f := f(a : b)) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x, u : A.x \wedge \neg A.u : u \sqsupseteq_{f(a:b).x} x) \\
\equiv & \quad \{\text{split off } x = a, \text{ using } A.a\} \\
 & (\forall x, u : A.x \ \wedge \ x \neq a \ \wedge \ \neg A.u : u \sqsupseteq_{f.x} x) \ \ \wedge \ \ (\forall u : \neg A.u : u \sqsupseteq_b a) \\
\equiv & \quad \{\text{first conjunct follows from } P2\} \\
 & (\forall u : \neg A.u : u \sqsupseteq_b a) \\
\equiv & \quad \{Q2\} \\
 & true \ ;
\end{array}
$$

$$
\begin{array}{ll}
 & Q0(f, b := f(a : b), f.a) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x : A.x : f(a : b).x \neq f.a) \\
\equiv & \quad \{\text{split off } x = a, \text{ using } A.a\} \\
 & (\forall x : A.x \wedge x \neq a : f.x \neq f.a) \ \ \wedge \ \ b \neq f.a \\
\equiv & \quad \{\text{first conjunct follows from } P0 \wedge A.a\} \\
 & b \neq f.a \\
\equiv & \quad \{Q0 \wedge A.a\} \\
 & true \ ;
\end{array}
$$

$$
\begin{array}{ll}
 & Q1(f, b := f(a : b), f.a) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x : A.x : x \sqsupseteq_{f.a} a \ \vee \ f.a \sqsupseteq_x f(a : b).x) \\
\equiv & \quad \{\text{split off } x = a, \text{ using } A.a\} \\
 & (\forall x : A.x \ \wedge \ x \neq a : x \sqsupseteq_{f.a} a \ \vee \ f.a \sqsupseteq_x f.x) \ \ \wedge \ \ (a \sqsupseteq_{f.a} a \ \vee \ f.a \sqsupseteq_a b) \\
\equiv & \quad \{\text{second conjunct follows from reflexivity of } \sqsupseteq_{f.a}\} \\
 & (\forall x : A.x \ \wedge \ x \neq a : x \sqsupseteq_{f.a} a \ \vee \ f.a \sqsupseteq_x f.x) \\
\equiv & \quad \{P1 \wedge A.a\} \\
 & true \ ;
\end{array}
$$

$$
\begin{array}{ll}
 & Q2(b := f.a) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x : \neg A.x : x \sqsupseteq_{f.a} a) \\
\equiv & \quad \{P2 \wedge A.a\} \\
 & true \ .
\end{array}
$$

We have now succeeded in showing that, in case $f.a \sqsupset_a b$, the swap $f.a, b := b, f.a$ respects the invariance of $P0..2 \wedge Q0..2 \wedge A.a$. The complete program thus becomes

$$
\begin{array}{l}
|[\ \textbf{var}\ A\colon \textbf{set of}\ X\,; \\
\quad A := \emptyset \\
\quad \{\text{inv}\ P0..2;\ \text{bd}\ |X \setminus A|\} \\
;\ \ \textbf{do}\ A \neq X \rightarrow |[\ \textbf{var}\ a\colon\ X;\ b\colon\ Y\,; \\
\qquad\qquad\qquad S2\ \{Q0\} \\
\qquad\qquad ;\ \ a := (\sqcap_b\, x :: x) \\
\qquad\qquad\quad \{\text{inv}\ P0..2 \wedge Q0..2\} \\
\qquad\qquad ;\ \ \textbf{do}\ A.a \rightarrow\ \ \textbf{if}\ f.a \sqsupseteq_a b \rightarrow f.a, b := b, f.a \\
\qquad\qquad\qquad\qquad\qquad\quad [\!]\ \ f.a \sqsubseteq_a b \rightarrow \mathit{skip} \\
\qquad\qquad\qquad\qquad\qquad\quad \textbf{fi} \\
\qquad\qquad\qquad\qquad\qquad\quad \{f.a \sqsubseteq_a b \wedge P0..2 \wedge Q0..2 \wedge A.a\} \\
\qquad\qquad\qquad\qquad\quad ;\ a := (\sqcap_b\, x : x \sqsupset_b a : x) \\
\qquad\qquad\quad \textbf{od} \\
\qquad\qquad ;\ \ f.a := b \\
\qquad\qquad ;\ \ A := A \cup \text{-}[a]\text{-} \\
\qquad\qquad]| \\
\quad \textbf{od} \\
]|\ .
\end{array}
$$

We postpone the coding of $S2$ until after the choice of a suitable data representation.

As the bound function previously given for the inner repetition is no longer appropriate, we have to reconsider termination of the inner repetition. We claim that

$$(\Sigma x : A.x : k.x.(f.x)) + k.a.b$$

is a suitable bound function, where

$$k.x.y = (\#u :: u \sqsupseteq_y x)\ . \tag{3}$$

The assignment to a clearly decreases the term $k.a.b$ and does not influence the others. It remains to show that the alternative statement does not increase the bound function.

$$
\begin{array}{ll}
 & ((\Sigma x : A.x : k.x.(f.x)) + k.a.b)(f, b := f(a : b), f.a) \\
= & \qquad \{\text{substitution}\} \\
 & (\Sigma x : A.x : k.x.(f(a : b).x)) + k.a.(f.a) \\
= & \qquad \{\text{split off } x = a, \text{ using } A.a\} \\
 & (\Sigma x : A.x\ \wedge\ x \neq a : k.x.(f.x)) + k.a.b + k.a.(f.a) \\
= & \qquad \{\text{merge domains of outer terms}\} \\
 & (\Sigma x : A.x : k.x.(f.x)) + k.a.b\ .
\end{array}
$$

6 An efficient program

The bound function given at the end of the preceding section shows the time complexity of the program to be $O(|X|^3)$. This is due to the fact that, in the inner repetition, the same elements are compared over and over again. The culprit seems to be the initialization $a := (\sqcap_b\, x :: x)$, which does not take into account any information gathered in previous steps. Inspired by (2),

we decide to remedy this by extending the invariants of both the inner and the outer repetition with

$$P3: \qquad (\forall x, y : y \sqsubset_x f.x \ \vee \ \neg A.x : x \sqsupseteq_y g.y) \ ,$$

where g is a variable of type **array** Y **of** X, initially satisfying

$$I0: \qquad (\forall y :: g.y = (\sqcap_y x :: x)) \ .$$

It follows from (2) that, if $P3$ holds, $Q1..2$ may be initialized by $a := g.b$.

Which program fragments are dangerous for the invariance of $P3$? There are two such fragments: the swap $f.a \sqsupset_a b \rightarrow f.a, b := b, f.a$ in the inner repetition and the final $f.a := b$; $A := A \cup \lbrace\!\lbrace a \rbrace\!\rbrace$ in the outer one. We show that neither of these actually damages $P3$:

$$\begin{array}{ll}
 & P3(f := f(a : b)) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x, y : y \sqsubset_x f(a : b).x \ \vee \ \neg A.x : x \sqsupseteq_y g.y) \\
\equiv & \quad \{\text{split off } x = a, \text{ using } A.a\} \\
 & (\forall x, y : x \neq a \ \wedge \ (y \sqsubset_x f.x \ \vee \ \neg A.x) : x \sqsupseteq_y g.y) \ \wedge \ (\forall y : y \sqsubset_a b : a \sqsupseteq_y g.y) \\
\equiv & \quad \{\text{first conjunct follows from } P3\} \\
 & (\forall y : y \sqsubset_a b : a \sqsupseteq_y g.y) \\
\Leftarrow & \quad \{P3 \text{ with } x := a\} \\
 & (\forall y : y \sqsubset_a b : y \sqsubset_a f.a) \\
\Leftarrow & \quad \{\text{transitivity}\} \\
 & f.a \sqsupseteq_a b \ ;
\end{array}$$

$$\begin{array}{ll}
 & P3(A := A \cup \lbrace\!\lbrace a \rbrace\!\rbrace)(f := f(a : b)) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x, y : y \sqsubset_x f(a : b).x \ \vee \ (\neg A.x \ \wedge \ x \neq a) : x \sqsupseteq_y g.y) \\
\equiv & \quad \{\text{split off } x = a\} \\
 & (\forall x, y : x \neq a \ \wedge \ (y \sqsubset_x f.x \ \vee \ \neg A.x) : x \sqsupseteq_y g.y) \ \wedge \ (\forall y : y \sqsubset_a b : a \sqsupseteq_y g.y) \\
\equiv & \quad \{\text{first conjunct follows from } P3\} \\
 & (\forall y : y \sqsubset_a b : a \sqsupseteq_y g.y) \\
\Leftarrow & \quad \{P3 \text{ with } x := a\} \\
 & \neg A.a \ .
\end{array}$$

So $P3$ is really an invariant of the program as it stands. However, if the introduction of g is to have any effect, we need to incorporate at least one statement to change the value of g. As the motivation for the introduction of g was to prevent repeated comparison of the same elements in the inner repetition, we propose to append $g.b := a$ to the body of the inner repetition. This respects the invariance of $P3$, as, with a' short for $(\sqcap_b x : x \sqsupset_b a : x)$, we have

$$\begin{array}{ll}
 & P3(g := g(b : a))(a := a') \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x, y : y \sqsubset_x f.x \ \vee \ \neg A.x : x \sqsupseteq_y g(b : a').y) \\
\equiv & \quad \{\text{split off } y = b\} \\
 & (\forall x, y : y \neq b \ \wedge \ (y \sqsubset_x f.x \ \vee \ \neg A.x) : x \sqsupseteq_y g.y) \ \wedge \ (\forall x : b \sqsubset_x f.x \ \vee \ \neg A.x : x \sqsupseteq_b a')
\end{array}$$

$$
\begin{array}{ll}
\equiv & \{\text{first conjunct follows from } P3\} \\
& (\forall x : b \sqsubset_x f.x \ \vee \ \neg A.x : x \sqsupseteq_b a') \\
\equiv & \{\text{definition of } a'\} \\
& (\forall x : b \sqsubset_x f.x \ \vee \ \neg A.x : x \sqsupset_b a) \\
\equiv & \{Q1..2, \text{ using } (2)\} \\
& (\forall x : b \sqsubset_x f.x \ \vee \ \neg A.x : x \neq a) \\
\equiv & \{\text{trading, one-point rule}\} \\
& b \sqsupseteq_a f.a \ \wedge \ A.a \ .
\end{array}
$$

We have now obtained the following program:

$$
\begin{array}{l}
\quad \{I0\} \\
|[\ \textbf{var}\ A\colon \textbf{set of}\ X\,; \\
\quad A := \emptyset \\
\quad \{\text{inv}\ P0..3;\ \text{bd}\ |X \setminus A|\} \\
;\ \ \textbf{do}\ A \neq X \rightarrow |[\ \textbf{var}\ a\colon X;\ b\colon Y\,; \\
\qquad\qquad\qquad\qquad S2\ \{Q0\} \\
\qquad\qquad\qquad ;\ \ a := g.b \\
\qquad\qquad\qquad\qquad \{\text{inv}\ P0..3 \wedge Q0..2\} \\
\qquad\qquad\qquad ;\ \ \textbf{do}\ A.a \rightarrow\ \textbf{if}\ f.a \sqsupset_a b \rightarrow f.a, b := b, f.a \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad [\!]\ \ f.a \sqsubseteq_a b \rightarrow \mathit{skip} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{fi} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{f.a \sqsubseteq_a b \wedge P0..3 \wedge Q0..2 \wedge A.a\} \\
\qquad\qquad\qquad\qquad\qquad\qquad ;\ a := (\sqcap_b\, x : x \sqsupset_b a : x) \\
\qquad\qquad\qquad\qquad\qquad\qquad ;\ g.b := a \\
\qquad\qquad\qquad\quad \textbf{od} \\
\qquad\qquad\qquad ;\ \ f.a := b \\
\qquad\qquad\qquad ;\ \ A := A \cup \{\!|a|\!\} \\
\qquad\qquad\qquad]| \\
\quad \textbf{od} \\
]|\ .
\end{array}
$$

This program is equivalent to the standard algorithm [2] [11]. As we shall see in the next section, its time complexity is $O(|X|^2)$, which is the best result attainable for this problem [10].

7 Complexity analysis

We claim that t, defined as

$$t = (\Sigma y :: k.(g.y).y) \ , \tag{4}$$

with k given by (3), is a suitable bound function for the inner repetition. The only statement that changes the value of t is $g.b := a$; if we manage to prove that this statement always decreases t, we may conclude that the algorithm as a whole has time complexity $O(|X|^2)$. Therefore, we now investigate whether

$$
\begin{array}{l}
\quad \{\, f.a \sqsubseteq_a b \;\wedge\; A.a \;\wedge\; P0..3 \;\wedge\; Q0..2 \;\wedge\; t = T \,\} \\
\quad a := (\sqcap_b\, x : x \sqsupset_b a : x) \\
; \; g.b := a \\
\quad \{\, t < T \,\} \quad .
\end{array}
$$

With a' short for $(\sqcap_b\, x : x \sqsupset_b a : x)$, we have

$$
\begin{array}{ll}
 & (t < T)(g := g(b : a))(a := a') \\
\equiv & \qquad \{(4), \text{substitution}\} \\
 & (\Sigma y :: k.(g(b : a').y).y) < T \\
\equiv & \qquad \{(4),\ t = T\} \\
 & (\Sigma y :: k.(g(b : a').y).y) < (\Sigma y :: k.(g.y).y) \\
\equiv & \qquad \{\text{split off } y = b, \text{ other terms cancel}\} \\
 & k.a'.b < k.(g.b).b \\
\equiv & \qquad \{(3)\} \\
 & (\#u :: u \sqsupseteq_b a') < (\#u :: u \sqsupseteq_b g.b) \\
\Leftarrow & \qquad \{\text{transitivity}\} \\
 & a' \sqsupset_b g.b \\
\Leftarrow & \qquad \{\text{definition of } a'\} \\
 & a = g.b \quad .
\end{array}
$$

The condition $a = g.b$ looks promising, as the invariant of the inner repetition can obviously be extended with

$$Q3 : \qquad a = g.b \quad .$$

Indeed, $Q3$ is established by the initialization $a := g.b$ and reestablished in each iteration step by $g.b := a$. However, we have not yet shown that $Q3$ still holds after the alternative statement. To this end, we observe

$$
\begin{array}{ll}
 & Q3(b := f.a) \\
\equiv & \qquad \{\text{substitution}\} \\
 & a = g.(f.a) \\
\Leftarrow & \qquad \{\, A.a \,\} \\
 & (\forall x : A.x : x = g.(f.x)) \quad .
\end{array}
$$

Thus, in order to discharge our proof obligation with respect to $Q3$, it is sufficient to show that

$$P4 : \qquad (\forall x : A.x : x = g.(f.x))$$

is an invariant of both repetitions. There are three program fragments that have a possible influence on $P4$: the swap and the assignment to $g.b$ in the inner repetition, and the final $f.a := b;\ A := A \cup \text{-}[a]\text{-}$ in the outer one. Taking them in order, we have

$$
\begin{array}{ll}
 & P4(f := f(a : b)) \\
\equiv & \qquad \{\text{substitution}\} \\
 & (\forall x : A.x : x = g.(f(a : b).x)) \\
\equiv & \qquad \{\text{split off } x = a, \text{ using } A.a\}
\end{array}
$$

$$
\begin{array}{ll}
 & (\forall x : A.x \ \wedge \ x \neq a : x = g.(f.x)) \ \wedge \ a = g.b \\
\equiv & \quad \{\text{first conjunct follows from } P4\} \\
 & a = g.b \\
\equiv & \quad \{Q3\} \\
 & true \ ;
\end{array}
$$

$$
\begin{array}{ll}
 & P4(g := g(b : a)) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x : A.x : x = g(b : a).(f.x)) \\
\equiv & \quad \{\text{split off terms with } f.x = b\} \\
 & (\forall x : A.x \ \wedge \ f.x \neq b : x = g.(f.x)) \ \wedge \ (\forall x : A.x \ \wedge \ f.x = b : x = a) \\
\equiv & \quad \{\text{first conjunct follows from } P4\} \\
 & (\forall x : A.x \ \wedge \ f.x = b : x = a) \\
\equiv & \quad \{Q0\} \\
 & true \ ;
\end{array}
$$

$$
\begin{array}{ll}
 & P4(A := A \cup \{\!|a|\!\})(f := f(a : b)) \\
\equiv & \quad \{\text{substitution}\} \\
 & (\forall x : A.x \ \vee \ x = a : x = g.(f(a : b).x)) \\
\equiv & \quad \{\text{split off } x = a, \text{ using } \neg A.a\} \\
 & (\forall x : A.x : x = g.(f.x)) \ \wedge \ a = g.b \\
\equiv & \quad \{\text{first conjunct is } P4\} \\
 & a = g.b \\
\equiv & \quad \{Q3\} \\
 & true \ .
\end{array}
$$

This concludes the proof of the algorithm's quadratic complexity.

8 Data refinement

The program description in the preceding sections is still at a very high level, namely, in terms of linear orderings. However, the complexity analysis depends on the assumption that statements like $a := (\sqcap_b\, x : x \sqsupset_b a : x)$ can be realized as $O(1)$-operations, thereby excluding the most naive implementations. It follows that we should pay some attention to the implementation of the program in terms of lower-level operations.

As a first step, we assume some numbering for the elements of Y is available and we identify the elements of Y with their numbers. In other words, we now assume $Y = [0..N)$ for some natural N.

In order to obtain an efficient coding for the statement $S2$ establishing $Q0$, we decide to strengthen the invariant of the outer repetition with

$$P5: \qquad (\forall y :: (\exists x : A.x : f.x = y) \ \equiv \ 0 \leq y < n)$$

and that of the inner repetition with

$$Q4: \qquad (\forall y :: (\exists x : A.x : f.x = y) \ \equiv \ 0 \leq y \leq n \ \wedge \ y \neq b) \ .$$

With these conventions, $S2$ can be simply coded as $b := n$. Moreover, the guard $A \neq X$ may equivalently be written as $n \neq N$. However, observe that $P5$ makes it necessary to extend the initialization $A := \emptyset$ with $n := 0$ and the increase $A := A \cup \{a\}$ with $n := n + 1$.

Finally, we come to the representations of the linear orderings. As can be seen from the program text, it is advantageous to represent the orderings of X and Y respectively in different ways. Assume that arrays $rank$ and $next$ are given that satisfy

$$I1: \quad (\forall x, y :: rank.x.y = (\#v :: v \sqsubset_x y)) \ ,$$
$$I2: \quad (\forall x, y :: next.x.y = (\sqcap_y u : u \sqsupset_y x : u)) \ .$$

(It is not important what the value of $next.x.y$ is in case x is the maximum of X with respect to $\sqsupseteq_y$.) Assuming $I1..2$, we may translate the condition $f.a \sqsupset_a b$ by $rank.a.(f.a) > rank.a.b$ and the assignment $a := (\sqcap_b x : x \sqsupset_b a : x)$ by $a := next.a.b$.

This gives the lower-level version of the program, in which all operations have been coded[4] in a way consistent with the assumptions in the complexity analysis:

```
   {I0..2}
|[ var n: int; A: set of X;
   A := ∅
;  n := 0
   {inv P0..5; bd |X \ A|}
;  do n ≠ N → |[ var a: X; b: Y;
                 b := n
              ;  a := g.b
                 {inv P0..4 ∧ Q0..4; bd (4)}
              ;  do A.a →  if rank.a.(f.a) > rank.a.b → f.a, b := b, f.a
                         [] rank.a.(f.a) ≤ rank.a.b → skip
                         fi
                         {f.a ⊑_a b  ∧  A.a ∧ P0..4 ∧ Q0..4}
                      ;  a := next.a.b
                      ;  g.b := a
                 od
              ;  f.a := b
              ;  A := A ∪ {a}
              ;  n := n + 1
              ]|
   od
]| .
```

Acknowledgements

I wish to thank the members of both the Eindhoven Tuesday Afternoon Club and the Eindhoven Algorithm Club, in particular Berry Schoenmakers, for their helpful criticism of previous versions of this paper.

[4] If desired, the set A may be replaced with a boolean array. We leave this to the reader.

References

[1] L. Allison, 'Stable marriages by coroutines'. *Inf. Proc. L.* **16** (1983), 61–65.

[2] D. Gale & L.S. Shapley, 'College admissions and the stability of marriage'. *Amer. Math. Monthly* **69** (1962), 9–15.

[3] D. Gale & M. Sotomayor, 'Ms. Machiavelli and the Stable Matching Problem'. *Amer. Math. Monthly* **92** (1985), 261–268.

[4] D. Gale & M. Sotomayor, 'Some remarks on the Stable Matching Problem'. *Discr. Appl. Math.* **11** (1985), 223–232.

[5] D. Gusfield, 'Three fast algorithms for four problems in stable marriage'. *SIAM J. Comp.* **16** (1987), 111–128.

[6] M.E.C. Hull, 'A parallel view of stable marriages'. *Inf. Proc. L.* **18** (1984), 63–66.

[7] R.W. Irving & P. Leather, 'The complexity of counting stable marriages'. *SIAM J. Comp.* **15** (1986), 655–667.

[8] R.W. Irving, P. Leather, & D. Gusfield, 'An efficient algorithm for the "optimal" stable marriage'. *J. ACM* **34** (1987), 532–543.

[9] D.G. McVitie & L.B. Wilson, 'The Stable Marriage Problem'. *Comm. ACM* **14** (1971), 486–492.

[10] C. Ng & D.S. Hirschberg, 'Lower bounds for the Stable Marriage Problem and its variants'. *SIAM J. Comp.* **19** (1990), 71–77.

[11] G. Pólya, R.E. Tarjan, & D.R. Woods, *Notes on introductory combinatorics*. Birkhäuser Verlag, Boston, 1983.

[12] N. Wirth, *Algorithms + data structures = programs*. Prentice-Hall, Englewood Cliffs, 1976.