

COMP4100 Lectures in 2007

by Malcolm Newey

EVACS in Z

Lecture 11 March 23, 2007

Scope: The total EVACS system (probably mentioned by Clive Boughton) has two parts – *vote collecting* and *vote counting*. This lecture is only about the types involved in the latter.

What are the Given Types

- PERSON?
- CANDIDATE?
- BALLOT?
- VOTE?
- PREFERENCE?
- PILE?
- PARTY?

Answers (Not a Slide)

- PERSON? No! Although each candidate is a person, no person other than candidates is relevant to the counting process
- CANDIDATE Yes! We'll clearly need to have variables of type CANDIDATE.
- BALLOT? Each ballot expresses a mapping from candidates to preferences.
- VOTE? No. When we say a candidate has 34892 votes, this is just a number associated her/him.
- PREFERENCE? No. A preference is just an integer.
- PILE? No. A pile of ballots is a structure (probably a bag).
- PARTY? No. It's because a candidates party doesn't affect the counting of votes.

Candidate Status

"Every candidate is marked in one of four ways:
CONTINUING, ELECTED, EXCLUDED, PENDING or BEING-EXCLUDED"

- The above constitutes a specification of an enumeration type; so:

$$\text{CandidateStatus} == \text{Continuing} \mid \text{Elected} \mid \text{Excluded} \\ \mid \text{Pending} \mid \text{BeingExcluded}$$
- Why is the following useful?

$$\text{CandidateStatusFunction} == \text{CANDIDATE} \rightarrow \text{CandidateStatus}$$

Information Content of a Ballot

Blah, blah, blah. Blah, blah, blah.	Blah, blah, blah. Etc. Etc..		
Ada	<input type="checkbox"/>	George	<input type="checkbox"/>
Adam	<input type="checkbox"/>	Giulia	<input type="checkbox"/>
Betty	<input type="checkbox"/>	Harry	<input type="checkbox"/>
Bill	<input type="checkbox"/>	Hope	<input type="checkbox"/>
Carol	<input type="checkbox"/>	Ian	<input type="checkbox"/>
Chris	<input type="checkbox"/>	Ingrid	<input type="checkbox"/>
Daisy	<input type="checkbox"/>	Jess	<input type="checkbox"/>
Donald	<input type="checkbox"/>	John	<input type="checkbox"/>
Edwin	<input type="checkbox"/>	Kate	<input type="checkbox"/>
Eve	<input type="checkbox"/>	Kevin	<input type="checkbox"/>

- 1 Bill
- 2 Daisy
- 3 Giulia
- 4 Hope
- 5 Adam
- 6 Chris

Ballots Specified

Since the information content of a ballot is a map from n of the candidates to (all of) the numbers 1 to n , we introduce a type *PreferenceList*.

$$PreferenceList == \{b : CANDIDATE \mapsto \mathbb{N}_1 \mid (max(\text{ran } b) = \#b) \bullet b\}$$

Since every every ballot paper contained in a ballot box denotes an object of this type, we define:

$$RawBallot == PreferenceList$$

and therefore we can introduce a (global) constant for the count:

$$\mid ballotBox : \text{bag } RawBallot$$

Ballots with Values

An *informal ballot* plays no part in the counting so has no value.

A vote that is assigned to a candidate because a surplus has been distributed has a reduced value.

Thus we can have introduce a more appropriate characterization of ballot objects:

<i>Ballot</i>
<i>prefs</i> : <i>PreferenceList</i>
<i>value</i> : \mathbb{R}
$\#prefs > 0$
$0 \leq value \wedge value \leq 1$

and use this type to specify what a pile of ballots is:

$$Pile == \text{bag } Ballot$$

Functions on Ballots

When all the candidates that have a preference on a ballot have been elected or excluded, that ballot is said to be *exhausted*.

$$exhausted : (Ballot \times CandidateStatusFunction) \rightarrow \mathbb{B}$$

$$\forall b : Ballot; statFun : CandidateStatusFunction \bullet exhausted(b, statFun) = \neg \exists c : CANDIDATE \bullet (c \in \text{dom } prefs(b) \wedge (statFun(c) = Continuing))$$

Functions on Piles - I

As the count proceeds in an ACT election ballots get sorted into a large number of piles according to commonalities between various preference functions exhibited by the ballots in those piles. Since each ballot has a value, the pile has a value.

$$\begin{array}{l}
 \text{pileValue} : \text{Pile} \rightarrow \mathbb{R} \\
 \text{pileValue}[\] = 0 \\
 \forall b : \text{Ballot} \bullet \text{pileValue}\{b \mapsto 1\} = \text{value}(b) \\
 \forall p1, p2 : \text{Pile} \bullet \text{pileValue}(p1 \uplus p2) = \text{pileValue}(p1) + \text{pileValue}(p2)
 \end{array}$$

Functions on Piles - II

If we have a pile of ballots, some which are exhausted, we often want to divide that pile into two – one of which holds the all exhausted ballots.

$$\begin{array}{l}
 \text{exhaustedBallots} : (\text{Pile} \times \text{CandidateStatusFunction}) \rightarrow \text{Pile} \\
 \forall p : \text{Pile}; \text{statFun} : \text{CandidateStatusFunction} \bullet \\
 \text{exhaustedBallots}(p, \text{statFun}) = \\
 \{ b : \text{Ballot}; n : \mathbb{N} \\
 \mid ((b \mapsto n) \in p) \wedge \text{exhausted}(b, \text{statFun}) \\
 \bullet (b \mapsto n) \}
 \end{array}$$

The State Schema

$$\begin{array}{l}
 \text{HCCA_State} \\
 \text{formals} : \text{Pile} \quad \text{[Computed once]} \\
 \text{quota} : \mathbb{N} \quad \text{[Computed once]} \\
 \text{count} : \mathbb{N} \\
 \text{exhaustedPiles} : \text{seq Pile} \\
 \\
 \text{candidateStatus} : \text{CandidateStatusFunction} \\
 \text{candidatePiles} : \text{CANDIDATE} \rightarrow \text{seq Pile} \\
 \text{candidatesCountWhenQuotaReached} : \text{CANDIDATE} \leftrightarrow \mathbb{N} \\
 \text{candidateTotals} : \text{CANDIDATE} \rightarrow \text{seq } \mathbb{N} \\
 \\
 \# \text{exhaustedPiles} = \text{count} \\
 \forall c : \text{CANDIDATE} \bullet \# \text{candidatePiles}(c) = \text{count} \\
 \forall c : \text{CANDIDATE} \bullet \# \text{candidateTotals}(c) = \text{count}
 \end{array}$$

The Obscure state variables

quota is the number of votes needed for a candidate to be elected.

count is incremented on each *surplus distribution* or *partial distribution* of votes when a candidate is being excluded.

candidatePiles: A sequence of piles of ballots, for each candidate, contains the votes that were acquired at the corresponding count.

candidatesCountWhenQuotaReached is, for each elected or pending candidate, the value of *count* which pertained when that candidate obtained a quota. For other candidates, it will be zero.

candidateTotals: As each candidate accumulates votes through both type of ballot redistribution, their progressive totals are recorded here.

Specification of the Algorithm

The operation of the HCCA system is one that has phases:

- As well as there being an initial state, there is an initialization phase;
- There is a phase of distributing first preferences;
- The main phase is counting which is iterative.

The three-phase aspect is specified using operation sequencing thus:

$$\text{HareClarke} \hat{=} \text{Initialization}; \text{DistributeFirstPreferences}; \text{MainLoop}$$

Specification of the Dynamics

The Hare-Clarke counting algorithm is interesting because it is largely iterative; most times around the main loop one candidate is eliminated and preferences are distributed. *MainLoop* has a body which is, in part:

$$\begin{aligned} \text{MainLoopBody} &\hat{=} \\ &\quad \text{ElectionOverSinceEnoughHaveQuota} \quad \vee \\ &\quad \text{ElectionOverSinceEnoughAreExcluded} \quad \vee \\ &\quad \text{ElectionNotOver} \quad \Rightarrow \\ &\quad \quad (\text{PendingCandidatesHaveQuotaExactly} \quad \Rightarrow \\ &\quad \quad \quad \text{ElectCandidatesHavingExactQuota} \quad \vee \\ &\quad \quad \dots) \end{aligned}$$