A Formal Analysis of the Will-Retire Correctness Statement

Nancy A. Day*, Mark D. Aagaard†, and Meng Lou‡

University of Waterloo Department of Computer Science Technical Report 2002-14

March, 2002

Abstract

We relate two microprocessor correctness statements and show that they are equivalent. The first correctness statement in question uses synchronization at retirement over a series of steps of the implementation and external equality as the required correspondence between states. The second correctness statement is the classic single-step commuting diagram with external equality as the match. We prove that if any microprocessor implementation and specification are shown to satisfy one of these correctness statements, they also satisfy the other correctness statement. This technical report is a continuation of Technical Report 2002-11 [DAL02] and includes little introductory material.

1 Introduction

In this paper, we describe a proof relating two commonly-used microprocessor correctness statements. One is a pointwise commuting diagram with external equivalence as the relation (match) between corresponding implementation and specification states. This correctness statement is similar to that of Burch and Dill [BD94] except it uses external equivalence to relate implementation and specification states rather than a flushing abstraction function. The second is a correctness statement based on synchronizations at retirement, where states are matched using external equivalence. This second correctness statement is used by Fox and Harman [FH00]. We show that under reasonable assumptions about the behaviour of microprocessors, these two correctness statements are equivalent for any implementation and specification machines. The machines may be non-deterministic. This result means that if any implementation and specification pair are shown to satisfy one of the correctness statements, they also satisfy the other correctness statement. Our result is for singlescalar implementations although we do not anticipate problems generalizing it for superscalar implementations, which may retire more than one instruction in a single step.

This technical report is a continuation of Technical Report 2002-11 [DAL02]. It is meant to be read after that report and therefore contains no introductory material on terminology or the Microbox framework [ACDJ01].

Figure 1 shows all the correctness statements currently described in the Microbox framework and the relationships between them. In all cases the missing last two letters in the name of the correctness statements are NN for non-deterministic implementation and specification. Details on the verification of the non-shaded relationships are described in Day *et al*. [DAL02]. The shaded area shows the two correctness statements and the relationship described in this paper.

In Sections 2 and 3, we describe the two correctness statements used in this paper. In Section 4, we describe the proof of the relationship between the two correctness statements.

2 Informed pointwise with equality

Pointwise alignment is the classic commuting diagram due to Milner [Mil71]. *Informed-pointwise* (I) (Definition 1) is a variation of pointwise alignment that allows the implementation to inform the correctness statement about how many

*Computer Science, University of Waterloo, nday@cs.uwaterloo.ca

†Electrical and Computer Engineering, University of Waterloo, markaa@swen.uwaterloo.ca

‡Computer Science, University of Waterloo, mlou@student.math.uwaterloo.ca

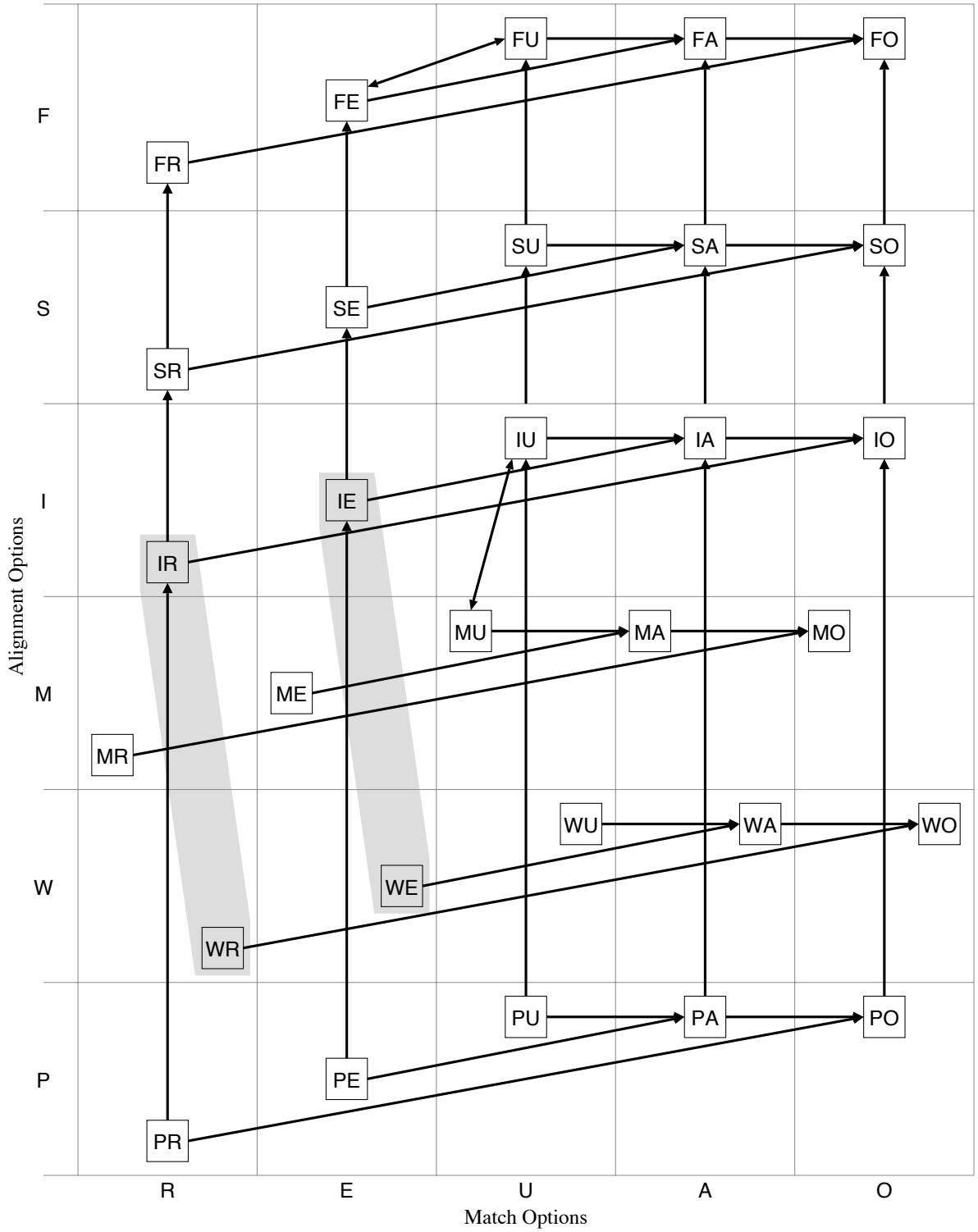
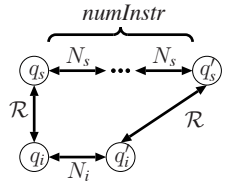


Figure 1: Partial order for alignment and match options

steps to take. If used with synchronization at fetch, the number of instructions relevant is the number fetched in the implementation step that will eventually retire. This accommodates pipeline stalls. For synchronization at retirement, which is our interest in this paper, the relevant number is how many instructions are retired (*numRetire*).

Definition 1 (Informed-pointwise induction clause: IONN)

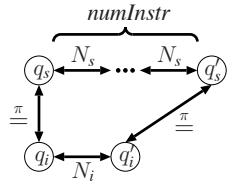
$$\begin{array}{l}\mathsf{IONN}(\mathit{numInstr}, \mathcal{R}, N_i, N_s) \equiv \\ \quad \forall\, q_i, q_i'.\ \forall\, q_s.\ \exists\, q_s'. \\ \qquad \text{let } j = \mathit{numInstr}(q_i, q_i') \text{ in} \\ \qquad\qquad \begin{bmatrix} \wedge & N_i(q_i, q_i') \\ & \mathcal{R}(q_i, q_s) \end{bmatrix} \implies \begin{bmatrix} \wedge & N_s^j(q_s, q_s') \\ & \mathcal{R}(q_i', q_s') \end{bmatrix}\end{array}$$



In this paper, we are concerned with the particular instance of informed pointwise alignment where $\mathcal{R}$ is external equivalence. We call this match equality (E). Definition 2 is the IENN correctness statement.

Definition 2 (Informed-pointwise induction clause: IENN)

$$\begin{array}{l}\mathsf{IONN}(\mathit{numInstr}, \mathcal{R}, N_i, N_s) \equiv \\ \quad \forall\, q_i, q_i'.\ \forall\, q_s.\ \exists\, q_s'. \\ \qquad \text{let } j = \mathit{numInstr}(q_i, q_i') \text{ in} \\ \qquad\qquad \begin{bmatrix} \wedge & N_i(q_i, q_i') \\ & q_i \overset{\pi}{=} q_s \end{bmatrix} \implies \begin{bmatrix} \wedge & N_s^j(q_s, q_s') \\ & q_i' \overset{\pi}{=} q_s' \end{bmatrix}\end{array}$$

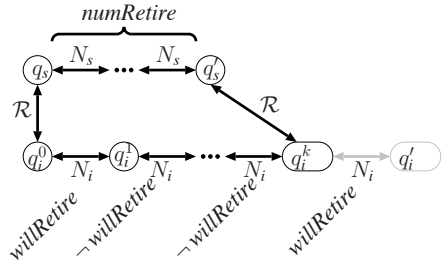


3 Will-retire with equivalence

In will-retire alignment (W) (Definition 3), the implementation retires an instruction in the first step of the trace. The implementation continues to take steps until it is ready to retire an instruction again. The implementation state just before an instruction is retired is matched against a specification state. The number of steps the specification takes depends on how many instructions are retired in the first implementation step.

Definition 3 (Will-retire induction clause: WONN)

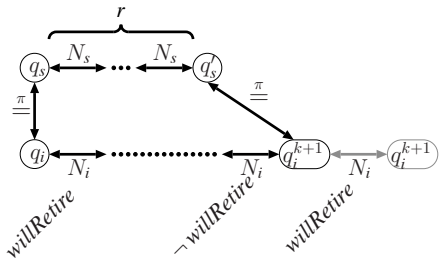
$$\begin{array}{l}\mathsf{WONN}(\mathit{numRetire}, \mathcal{R}, N_i, N_s) \equiv \\ \quad \forall\, q_i^0, q_i^1, \ldots, q_i^k.\ \forall\, q_s.\ \exists\, q_s'. \\ \qquad \text{let } r = \mathit{numRetire}(q_i^0, q_i^1) \textit{ in} \\ \qquad\qquad \begin{bmatrix} & N_i(q_i^0, q_i^1) \\ \wedge & \mathit{willRetire}(q_i^0, q_i^1) \\ \wedge & (\ \forall j \in 1 \ldots k-1.\ N_i(q_i^j, q_i^{j+1})\ \wedge\ \neg\, \mathit{willRetire}(q_i^j, q_i^{j+1})\ ) \\ \wedge & (\ \exists\, q_i.\ N_i(q_i^k, q_i')\ \wedge\ \mathit{willRetire}(q_i^k, q_i')\ ) \\ \wedge & \mathcal{R}(q_i^0, q_s) \end{bmatrix} \implies \begin{bmatrix} \wedge & N_s^r(q_s, q_s') \\ & \mathcal{R}(q_i^k, q_s') \end{bmatrix}\end{array}$$



Because will-retire alignment involves synchronization at retirement, it can be combined with the equality match rather than a flushing abstraction. Will-retire with the equality match (WENN, Definition 4) is an instance of WONN with external equivalence ($q_i \overset{\pi}{=} q_s$) as the match between specification and implementation states. The implementation projection function (π_i) must associate the program counter in the specification with the address of the next instruction to be retired.

Definition 4 (Will-retire with equality induction clause: WENN)

$$
\begin{array}{l}
\mathsf{WONN}(\mathit{numRetire}, \mathcal{R}, N_i, N_s) \equiv \\
\quad \forall\, q_i^0, q_i^1, \ldots, q_i^k.\ \forall\, q_s.\ \exists\, q_s'. \\
\quad\quad \mathit{let}\ r = \mathit{numRetire}(q_i^0, q_i^1)\ \mathit{in} \\
\quad\quad\quad \left[\begin{array}{ll}
 & N_i(q_i^0, q_i^1) \\
\wedge & \mathit{willRetire}(q_i^0, q_i^1) \\
\wedge & (\ \forall j \in 1 \ldots k-1.\ N_i(q_i^j, q_i^{j+1})\ \wedge\ \neg\, \mathit{willRetire}(q_i^j, q_i^{j+1})\ ) \\
\wedge & (\ \exists\, q_i.\ N_i(q_i^k, q_i')\ \wedge\ \mathit{willRetire}(q_i^k, q_i')\ ) \\
\wedge & q_i^0 \overset{\pi}{=} q_s
\end{array} \right] \Longrightarrow \left[\begin{array}{ll} \wedge & N_s^r(q_s, q_s') \\ & q_i^k \overset{\pi}{=} q_s' \end{array} \right]
\end{array}
$$



4 Proof

We prove that will-retire equality (WENN, Definition 4) is equivalent to informed-pointwise with equality (IENN, Definition 2). The first insight in the proof of WENN $\iff$ IENN is the introduction of an alternative way of expressing WONN that appears to be a tighter correctness criteria, but is actually equivalent to WONN with a reasonable assumption about the relationship between the predicate *willRetire* and the function *numRetire*. We call this equivalent formulation single-step will-retire (ssWONN, Definition 5). ssWONN decomposes WONN into two simpler, single-step, properties based on whether the implementation will retire an instruction.

Definition 5 (Single-step will-retire induction clause: ssWONN)

$$
\begin{array}{l}
\mathsf{ssWONN}(\mathit{willRetire}, \mathcal{R}, N_i, N_s) \equiv \\
\quad \forall\, q_i, q_i', q_s. \\
\quad\quad \left[\begin{array}{l} N_i(q_i, q_i') \\ \mathcal{R}(q_i, q_s) \end{array} \right] \Longrightarrow \left[\begin{array}{lrcl} \wedge & \mathit{willRetire}(q_i, q_i') & \Longrightarrow & \exists\, q_s'.\ N_s(q_s, q_s') \wedge \mathcal{R}(q_i', q_s') \\ & \neg\, \mathit{willRetire}(q_i, q_i') & \Longrightarrow & \mathcal{R}(q_i', q_s) \end{array} \right]
\end{array}
$$

Single-step will-retire is similar to informed-pointwise (IONN, Definition 1) in examining only a single step of the implementation. We prove the two correctness statements are equivalent under Condition 1, the *willretire_cond*, (Theorem 1).

Theorem 1 (IONN $\iff$ ssWONN)

$$
\begin{array}{l}
\forall\ \mathit{willRetire}, \mathit{numRetire}, \mathcal{R}, N_i, N_s. \\
\quad \mathit{willretire\_cond}(\mathit{numRetire}, \mathit{willRetire}) \Longrightarrow \qquad \text{— Condition 1} \\
\quad\quad (\ \mathsf{IONN}(\mathit{numRetire}, \mathcal{R}, N_i, N_s) \iff \mathsf{ssWONN}(\mathit{willRetire}, \mathcal{R}, N_i, N_s)\ )
\end{array}
$$

Condition 1 states that if *willRetire* is true of a step then *numRetire* is one, otherwise it is zero.

Condition 1 (*willRetire* and *numRetire*)

$$
\begin{array}{l}
\mathit{willretire\_cond}(\mathit{numRetire}, \mathit{willRetire}) \equiv \\
\quad \forall\, q_i, q_i'.\ \mathit{numRetire}(q_i, q_i') = \text{if}\ \mathit{willRetire}(q_i, q_i')\ \text{then}\ 1\ \ \text{else}\ 0
\end{array}
$$

By specialized $\mathcal{R}$ in Theorem 1 to the equality match, we are able to conclude IENN is equivalent to ssWENN as long as Condition 1 holds.

The next and more challenging step in the proof is to show that will-retire with the equality match is equivalent to single-step will-retire (WENN $\iff$ ssWENN). Showing ssWENN $\Longrightarrow$ WENN is straightforward by induction. The other direction of showing WENN $\Longrightarrow$ ssWENN, stated in Theorem 2, holds under the conditions listed. Figure 2 is

an illustration of the proof of Theorem 2. In Step 0, we start with the left and lower side of the commuting diagram for ssWENN.

In Step 1, we use a liveness condition, called the eventually retires condition (Condition 2), to reach a future state, $q_i^\star$, that retires an instruction. Condition 2 states that from any implementation state it is always possible to reach a state that will retire an instruction.

Condition 2 (Eventually Retires)

$$\begin{aligned}&\mathit{eventually\_retires}(\mathit{willRetire}, N_i) \equiv \\ &\quad \forall\, q_i.\ \exists\, k.\ q_i^0, \ldots, q_i^k, q_i^{k+1}. \\ &\qquad q_i = q_i^0 \ \wedge\ (\forall\, j < k.\ N_i(q_i^j, q_i^{j+1}) \ \wedge\ \neg\, \mathit{willRetire}(q_i^j, q_i^{j+1})) \\ &\qquad \wedge\ N_i(q_i^k, q_i^{k+1}) \ \wedge\ \mathit{willRetire}(q_i^k, q_i^{k+1})\end{aligned}$$

In Step 3, we use the π_i*_no_retire_cond* condition (Condition 3) to conclude the projection of q_i' and $q_i^\star$ are equal. Condition 3 relates the predicate *willRetire* to the implementation projection function π_i. It says that if an instruction is not retired in a step where the implementation transitions from q_i to q_i', then the projections of q_i and q_i' are equivalent. This condition assumes that π_i is appropriate for synchronization at retirement.

Condition 3 (*willRetire* and π_i)

$$\begin{aligned}&\pi_i\mathit{\_no\_retire\_cond}(\mathit{willRetire}, \pi_i, N_i) \equiv \\ &\quad \forall\, q_i, q_i'.\ \neg\, \mathit{willRetire}(q_i, q_i') \wedge N_i(q_i, q_i') \implies \pi_i(q_i) = \pi_i(q_i')\end{aligned}$$

In Step 4, we use WENN to complete the commuting diagram. Step 5 shows WENN where the left case follows from Step 3 and the right case follows directly from Condition 3.

Theorem 2 (WENN $\implies$ ssWENN)

$$\begin{aligned}&\forall\ \mathit{willRetire}, \pi_i, \pi_s, N_i, N_s. \\ &\quad \left[\begin{array}{cll} \wedge & \pi_i\mathit{\_no\_retire\_cond}(\mathit{willRetire}, \pi_i, N_i) & \text{— Condition 3} \\ & \mathit{eventually\_retires}(\mathit{willRetire}) & \text{— Condition 2} \end{array}\right] \\ &\quad \implies (\ \mathsf{WENN}(\mathit{willRetire}, \pi_i, \pi_s, N_i, N_s) \implies \mathsf{ssWENN}(\mathit{willRetire}, \pi_i, \pi_s, N_i, N_s)\ )\end{aligned}$$

Theorem 3 (WENN $\iff$ IENN)

$$\begin{aligned}&\forall\ \mathit{willRetire}, \pi_i, \pi_s, N_i, N_s. \\ &\quad \left[\begin{array}{cll} & \pi_i\mathit{\_no\_retire\_cond}(\mathit{willRetire}, \pi_i, N_i) & \text{— Condition 3} \\ \wedge & \mathit{eventually\_retires}(\mathit{willRetire}, N_i) & \text{— Condition 2} \\ \wedge & \mathit{willretire\_cond}(\mathit{numRetire}, \mathit{willRetire}) & \text{— Condition 1} \end{array}\right] \\ &\quad \implies (\ \mathsf{WENN}(\mathit{willRetire}, \pi_i, \pi_s, N_i, N_s) \iff \mathsf{IENN}(\mathit{numRetire}, \pi_i, \pi_s, N_i, N_s)\ )\end{aligned}$$

Combining Theorems 1 and 2 and ssWENN $\implies$ WENN, we can conclude WENN is equivalent to IENN under the conditions listed (Theorem 3). Using previous results, where it was shown that IENN implies FENN [DAL02], we conclude that WENN implies FENN (flushpoint alignment with the equality match).

The match relation R ($\mathit{abs}(q_i) = q_s) \wedge q_i \overset{\pi}{=} q_s$) reduces to the match E when the specification has no internal state (i.e. $\pi_s = \mathit{id}$). Therefore we can also conclude that WRNN $\iff$ IRNN, when the specification has no internal state.

The proof presented in this section is similar in flavour to a proof relating informed-pointwise flushing (IUNN) with the must-issue correctness statement where the implementation taking one step where it fetches an instruction followed by some number of stalled steps. Details on this proof can be found in Day *et al*. [DAL02].

5 Conclusions

In this paper, we described a proof of the equivalence of the following two microprocessor correctness statements: informed pointwise alignment and will-retire alignment both with the equality match. This result means that if any implementation and specification pair are shown to satisfy one of the correctness statements, they also satisfy the other correctness statement. This proof was mechanized in the HOL theorem prover [GM93].

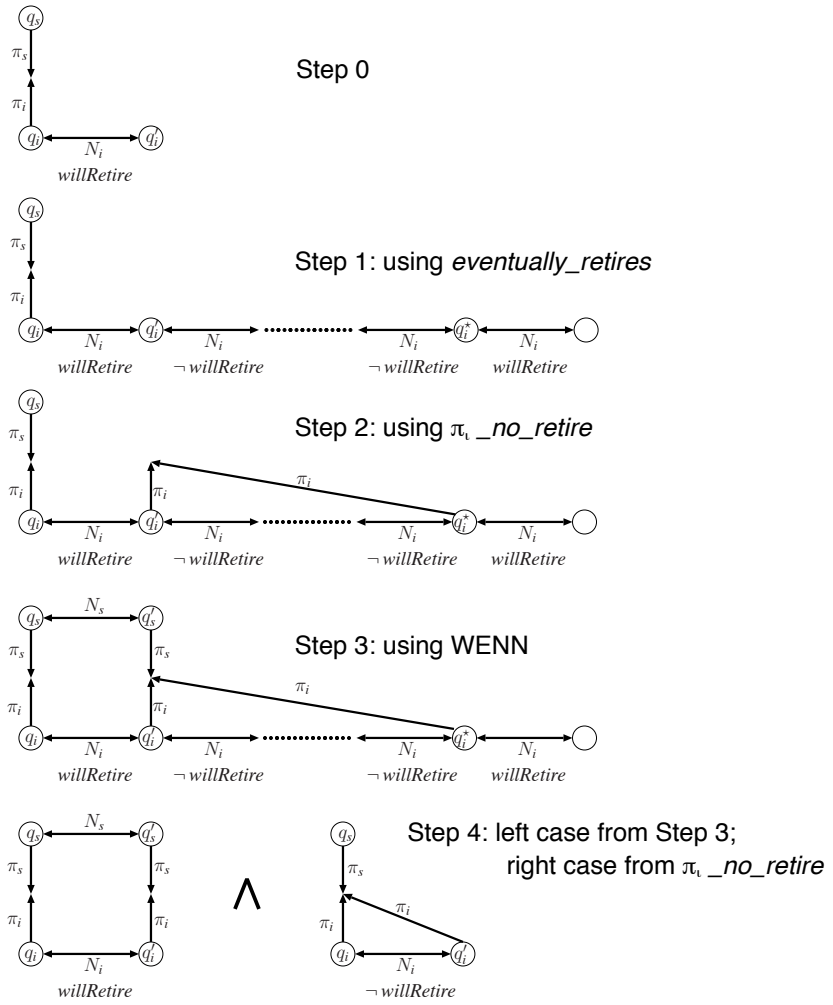


Figure 2: Steps in proof of WENN $\Longrightarrow$ ssWENN

Acknowledgments

We thank Byron Cook of Prover Technologies, and Robert Jones of Intel for early discussions on this topic. The authors are supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). Aagaard is supported in part by Intel Corporation.

References

[ACDJ01] M. D. Aagaard, B. Cook, N. A. Day, and R. B. Jones. A framework for microprocessor correctness statements. In *CHARME*, volume 2144 of *LNCS*, pages 433–448. Springer, 2001.

[BD94] J. Burch and D. Dill. Automatic verification of pipelined microprocessor control. In D. L. Dill, editor, *CAV*, volume 818 of *LNCS*, pages 68–80. Springer Verlag; New York, 1994.

[DAL02] N. A. Day, M. D. Aagaard, and M. Lou. A mechanized theory for microprocessor correctness statements. Technical Report 2002-11, University of Waterloo, Department of Computer Science, 2002.

[FH00] A. Fox and N. Harman. Algebraic models of correctness for microprocessors. *Formal Aspects in Computing*, 12(4):298–312, 2000.

[GM93] M. Gordon and T. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Joint Conference on Artificial Intelligence*, pages 481–489, 1971.