Decomposing Properties into Safety and Liveness using Predicate Logic†

Fred B. Schneider

87-874

October 1987

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

†This material is based on work supported in part by the Office of Naval Research under contract N00014-86-K-0092 and the National Science Foundation under Grant No. CCR-8701103. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not reflect the views of the Office of Naval research or National Science Foundation.

Decomposing Properties into Safety and Liveness using Predicate Logic*

Fred B. Schneider

Department of Computer Science
Cornell University
Ithaca, New York 14853

October 5, 1987

ABSTRACT

A new proof is given that every property can be expressed as a conjunction of safety and liveness properties. The proof is in terms of first-order predicate logic.

*This material is based on work supported in part by the Office of Naval Research under contract N00014-86-K-0092 and the National Science Foundation under Grant No. CCR-8701103. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not reflect the views of the Office of Naval Research or National Science Foundation.

1. Introduction

Two classes of properties are of particular interest when considering programs: safety properties and liveness properties. Informally, a *safety property* stipulates that "bad things" do not happen during execution of a program and a *liveness property* stipulates that "good things" do happen (eventually) [2]. Distinguishing between safety and liveness properties is useful because knowing whether a property is safety or liveness helps when deciding how to prove that the property holds for a program.

In [1], formal definitions of safety and liveness are given and it is proved that every property can be expressed as the conjunction of a safety property and a liveness property. The formal definitions of safety and liveness are given in terms of first-order predicate logic, but the proof that every property can be decomposed into safety and liveness is not—it uses topology. The purpose of this paper is to give a proof of this theorem using only first-order predicate logic.

2. Specifying Properties

A *program state* is a mapping from variables to values. An execution of a concurrent program can be viewed as an infinite sequence of program states

$$\sigma = s_0 \, s_1 \, \ldots,$$

which we call a *history*. In a history, s_0 is an initial state of the program and each subsequent state results from executing a single atomic action in the preceding state. (For a terminating execution, an infinite sequence is obtained by repeating the final state.) A *property* is a set of such sequences.

One way to specify a property is by using first-order predicate logic. For a state s, define $s.v$ to be the value of variable v in that state. A formula of first-order predicate logic where s is the only free variable defines a set of states. For example,

$$(\forall i\colon\ 1 \le i < N\colon\ s.a[i] \le s.a[i+1])$$

specifies the set of states in which the elements of array $a[1{:}N]$ are sorted. Usually "$s.$" is implicit and therefore left out of such a formula, resulting in the more familiar use of first-order predicate logic as an assertion language.

A set of sequences of states—a property—can also be defined using first-order predicate logic. To facilitate such specifications, for any sequence $\sigma = s_0 \, s_1 \ldots$ define for $0 \le i$:

$\sigma[i] \;\equiv\; s_i$.
$\sigma[..i] \;\equiv\; s_0 \, s_1 \ldots s_{i-1}$. The empty sequence if $i=0$.
$|\sigma| \;\equiv\;$ the length of σ (ω if σ is infinite).

A formula of first-order predicate logic in which σ is the only free variable defines the set of sequences that satisfy the formula and therefore specifies a property. For example,

$$(\forall i\colon\ 0 \le i\colon\ \sigma[i].v = 0)$$

specifies the property in which the value of v remains 0 throughout execution.

We write $\alpha \models P$ if $\alpha \in S^\omega$ is in the property specified by P. Thus,

$$\alpha \models P \;=\; P^\sigma_\alpha.$$
$$\alpha \not\models P \;=\; \neg P^\sigma_\alpha.$$

3. Safety and Liveness

According to [1], a property P is a safety property provided

$$\text{Safety: } (\forall \sigma:\ \sigma \in S^\omega:\ \sigma \not\models P \Rightarrow (\exists i:\ 0 \le i:\ (\forall \beta:\ \beta \in S^\omega:\ \sigma[..i]\beta \not\models P))), \tag{3.1}$$

where S is the set of program states, S^* the set of finite sequences of states, S^ω the set of infinite sequences of states, and juxtaposition is used to denote catenation of sequences. A property P is a liveness property provided

$$\text{Liveness: } (\forall \alpha:\ \alpha \in S^*:\ (\exists \beta:\ \beta \in S^\omega:\ \alpha\beta \models P)). \tag{3.2}$$

Given a property P, we are interested in defining properties $Safe(P)$ and $Live(P)$ such that

- $Safe(P)$ is a safety property,
- $Live(P)$ is a liveness property, and
- $P = Safe(P) \wedge Live(P)$.

Observe that if

$$Safe(P) = P \vee M_P$$
$$Live(P) = P \vee \neg M_P$$

then

$$\begin{aligned} Safe(P) \wedge Live(P) &= (P \vee M_P) \wedge (P \vee \neg M_P) \\ &= (P \wedge P) \vee (P \wedge \neg M_P) \vee (M_P \wedge P) \vee (M_P \wedge \neg M_P) \\ &= P \end{aligned}$$

Hence, we have only to look for an M_P that makes $P \vee M_P$ (i.e. $Safe(P)$) a safety property and $P \vee \neg M_P$ (i.e. $Live(P)$) a liveness property.

It turns out that using

$$M_P:\ (\forall i:\ 0 \le i:\ (\exists \beta:\ \beta \in S^\omega:\ \sigma[..i]\beta \models P))$$

suffices. First, we show formally that $Safe(P)$ satisfies definition (3.1) of safety. The proof that follows is a sequence of first-order predicate logic formulas with explanations interspersed (and delimited by « and ») of how each formula is derived from its predecessor.

Choose any $\sigma \in S^\omega$:

$$\sigma \not\models Safe(P)$$

«by definition of *Safe*(P)»

$= \sigma \not\models (P \vee (\forall i: 0 \le i: (\exists \beta: \beta \in S^{\omega}: \sigma[..i]\beta \models P)))$

«by definition of $\not\models$»

$= \neg(P \vee (\forall i: 0 \le i: (\exists \beta: \beta \in S^{\omega}: \sigma[..i]\beta \models P)))^{\sigma}_{\sigma}$

«by substitution»

$= \neg(P \vee (\forall i: 0 \le i: (\exists \beta: \beta \in S^{\omega}: \sigma[..i]\beta \models P)))$

«by De Morgan's Laws»

$= \neg P \wedge (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P))$

«$A \wedge B \Rightarrow B$»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P))$

«because $(\forall x:: A) = (\forall x:: A \wedge (\forall y:: A^{x}_{y}))$»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P \wedge (\forall \gamma: \gamma \in S^{\omega}: \sigma[..i]\gamma \not\models P)))$

«because *true* $\wedge P = P$ and $(\sigma[..i]\beta)[..i] = \sigma[..i]$»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P \wedge (i=i) \wedge (\forall \gamma: \gamma \in S^{\omega}: (\sigma[..i]\beta)[..i]\gamma \not\models P)))$

«by substitution»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P \wedge (k=i)^{k}_{i} \wedge (\forall \gamma: \gamma \in S^{\omega}: (\sigma[..i]\beta)[..k]\gamma \not\models P)^{k}_{i}))$

«by $\exists$-Generalization»

$\Rightarrow (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P \wedge (\exists k: k=i: (\forall \gamma: \gamma \in S^{\omega}: (\sigma[..i]\beta)[..k]\gamma \not\models P))))$

«by Range Widening»

$\Rightarrow (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P \wedge (\exists k: 0 \le k: (\forall \gamma: \gamma \in S^{\omega}: (\sigma[..i]\beta)[..k]\gamma \not\models P))))$

«by De Morgan's Law»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P \wedge \neg(\forall k: 0 \le k: (\exists \gamma: \gamma \in S^{\omega}: (\sigma[..i]\beta)[..k]\gamma \models P))))$

«by definition of $\not\models$»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P \wedge \sigma[..i]\beta \not\models (\forall k: 0 \le k: (\exists \gamma: \gamma \in S^{\omega}: \sigma[..k]\gamma \models P))))$

«because $\alpha \not\models A \wedge \alpha \not\models B = \alpha \not\models (A \vee B)$»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models (P \vee (\forall k: 0 \le k: (\exists \gamma: \gamma \in S^{\omega}: \sigma[..k]\gamma \models P)))))$

«by definition of *Safe*(P)»

$= (\exists i: 0 \le i: (\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models Safe(P)))$

It is not surprising that *Safe*(P) is a safety property. If $\sigma \not\models Safe(P)$ then, by definition, $\sigma \not\models M_P$. However, this means there exists an i such that

$(\forall \beta: \beta \in S^{\omega}: \sigma[..i]\beta \not\models P)$.

We could consider prefix $\sigma[..i]$ to be a "bad thing". Thus, σ violates a safety property whenever $\sigma \not\models Safe(P)$.

We now show formally that *Live*(P) satisfies definition (3.2) of liveness.

$(\forall \alpha: \alpha \in S^{*}: true)$

«since *true* $= A \vee \neg A$»

$= (\forall \alpha: \alpha \in S^{*}: (\exists \beta: \beta \in S^{\omega}: \alpha\beta \models P) \vee \neg(\exists \beta: \beta \in S^{\omega}: \alpha\beta \models P))$

«renaming bound variable β to γ»

$= (\forall \alpha: \alpha \in S^{*}: (\exists \beta: \beta \in S^{\omega}: \alpha\beta \models P) \vee \neg(\exists \gamma: \gamma \in S^{\omega}: \alpha\gamma \models P))$

«since β is not free in $(\exists \gamma: \gamma \in S^{\omega}: \alpha\gamma \models P)$»

$= (\forall \alpha: \alpha \in S^{*}: (\exists \beta: \beta \in S^{\omega}: \alpha\beta \models P \vee \neg(\exists \gamma: \gamma \in S^{\omega}: \alpha\gamma \models P)))$

«by De Morgan's Law»

$= (\forall \alpha: \alpha \in S^{*}: (\exists \beta: \beta \in S^{\omega}: \alpha\beta \models P \vee (\forall \gamma: \gamma \in S^{\omega}: \alpha\gamma \not\models P)))$

«since $true \wedge A = A$»

$= (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models P \vee (|\alpha|=|\alpha| \wedge (\forall\gamma\colon \gamma\in S^\omega\colon \alpha\gamma\not\models P))))$

«by substitution, since $(\alpha\beta)[..|\alpha|]=\alpha$»

$= (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models P \vee ((i=|\alpha|)^i_{|\alpha|} \wedge (\forall\gamma\colon \gamma\in S^\omega\colon (\alpha\beta)[..i]\gamma\not\models P)^i_{|\alpha|})))$

«by $\exists$-Generalization»

$\Rightarrow (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models P \vee (\exists i\colon i=|\alpha|\colon (\forall\gamma\colon \gamma\in S^\omega\colon (\alpha\beta)[..i]\gamma\not\models P))))$

«by Range Widening»

$\Rightarrow (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models P \vee (\exists i\colon 0\le i\colon (\forall\gamma\colon \gamma\in S^\omega\colon (\alpha\beta)[..i]\gamma\not\models P))))$

«by De Morgan's Law»

$= (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models P \vee \neg(\forall i\colon 0\le i\colon (\exists\gamma\colon \gamma\in S^\omega\colon (\alpha\beta)[..i]\gamma\models P))))$

«by definition of $\alpha\beta\models A$»

$= (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models P \vee \alpha\beta\models\neg(\forall i\colon 0\le i\colon (\exists\gamma\colon \gamma\in S^\omega\colon \sigma[..i]\gamma\models P))))$

«because $\alpha\beta\models A \vee \alpha\beta\models B = \alpha\beta\models(A\vee B)$»

$= (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models (P \vee\neg(\forall i\colon 0\le i\colon (\exists\gamma\colon \gamma\in S^\omega\colon \sigma[..i]\gamma\models P))))$

«by definition of $Live(P)$»

$= (\forall\alpha\colon \alpha\in S^*\colon (\exists\beta\colon \beta\in S^\omega\colon \alpha\beta\models Live(P)))$

«by Liveness definition (3.2)»

$=$ $Live(P)$ is liveness.

An informal justification that $Live(P)$ is liveness is the following. If $\sigma\not\models Live(P)$ then, by definition, $\sigma\models M_P$. From, $\sigma\models M_P$, we conclude that it always remains possible for some "good thing" (i.e. β in M_P) to happen. This is the defining characteristic of liveness, so σ violates a liveness property whenever $\sigma\not\models Live(P)$.

Acknowledgment

David Gries made numerous suggestions—some of which I even adopted—about presenting the proofs.

References

[1] Alpern, B., and F.B. Schneider. Defining liveness. *Information Processing Letters 21* (Oct. 1985), 181-185.

[2] Lamport, L. Proving the correctness of multiprocess programs. *IEEE Trans. on Software Engineering SE-3*, 2 (March 1977), 125-143.