A Normal Form for Multi-Exit Statements

Emil Sekerinski
Department of Computing and Software
McMaster University
Hamilton, ON, Canada
emil@mcmaster.ca

Tian Zhang
Department of Computing and Software
McMaster University
Hamilton, ON, Canada
zhangt26@mcmaster.ca

For a class of statements, a normal form provides a uniform way for their formal specification. Control structures like exception handling introduce additional exit(s) of statements, and new normal forms as a consequence. In this paper, we give the normal forms of several classes of multi-exit statements, explore their algebraic properties, and discuss their potential of being utilized for the development of programs, where a postcondition for each exit is required. All the theorems have been checked with a formal verification tool.

1 Introduction

Normal forms of programs are of interest for a variety of reasons. Hoare et al. use a normal form to show the completeness of algebraic laws of straight-line programs with bounded nondeterminism [10]; since refinement can be expressed by equality and nondeterministic choice, this shows also the completeness of refinement laws. Hennessy and Milner as well as Roscoe and Hoare consider normal forms of programs with synchronous communication as a way of defining those [9, 20]. Jifeng, Page, and Bowen study the transformation of occam programs into a normal form suitable for FPGA implementation [13]. Hoare, Jifeng, and Sampaio express compilation of guarded command programs as a transformation to a normal form [11]. Von Wright presents a normal form of programs with demonic and angelic (unbounded) non-determinism [25]; the normal form relies on (abstract) angelic and demonic *update statements*. Abrial gives a normal form for programs with demonic nondeterminism, which justifies the specification constructs of the B method [1]. Kozen proves that every program with while-loops and bounded nondeterminism can be transformed into a normal form with a single loop using only the rules of Kleene algebra with tests [15]. Back and von Wright treat systematically normal forms for various classes of programs, including those with demonic and angelic nondeterminism [3]. Ying uses the framework of Back and von Wright for a normal form of probabilistic programs [26]. Borba et al. propose a normal form for object-oriented programs [6], with application to refactoring. Silva, Sampaio, and Barros use a normal form of occam programs for hardware / software partitioning [23]. Li, Zhu and He use a normal form for a language with compensable transactions [17].

In the present work, we study normal forms for statements with a *single entry* and *multiple exits*. Such statements arise in programming languages that allow returning from inside a procedure, breaking a loop, exiting from inside a program, or raising exceptions. These control structures offer flexible manipulations of control flow. For example, the exception handling models in Fig. 1 result multi-exit control structures: the termination model in Java has a normal exit and an exceptional exit; the retry model in Eiffel [18] allows **retry** statements that re-execute the body after dealing with exceptions, resulting an additional retry exit that always redirects the control flow to the beginning of method.

We consider sequential programs and use the predicate transformer model for statements, as it is expressive enough to distinguish between blocking and abortion, and includes both demonic and angelic

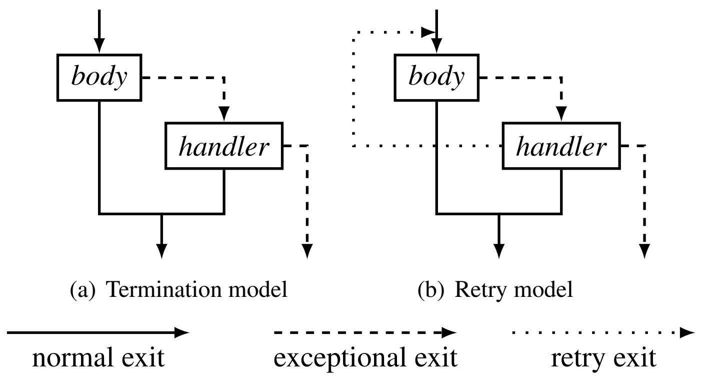

Figure 1: Two Exception Handling Mechanisms

Java exception handling

```
try {
  S
}
catch (Exception e) {
  T
}
```

Eiffel exception handling

```
M
  require
    P
  local
    vars
  do
    S
  ensure
    Q
  rescue
    T
    [retry]
  end
```

Figure 2: Java / Eiffel Exception Handling Code

nondeterminism. An interpretation of demonic and angelic nondeterminism is by games, e.g. [3].

Cristian defines statements with single entry and multiple exits as a set of predicate transformers [7]. As King and Morgan point out, this disallows nondeterminism [14]; the solution is to use a single predicate transformer with one postcondition for each exit instead, which we follow here. Jacobs formalizes multi-exit statements of the Java language using state transformers, which again precludes nondeterminism [12]. Leino and Snepscheut study algebraic properties and derive weakest preconditions of statements from a trace semantics. Here we start with a weakest preconditions semantics, from which algebraic properties are derived.

We formalize the semantics of multi-exit statements in higher-order logic, following the methodology of [8, 4, 2]. An intuitive means to formalize multi-exit statements is to encode the choice of exit as an additional variable, then the theorems of traditional single-exit statements become applicable. However, we chose to directly formalize these statements as higher-order predicate transformers for two reasons. First, different state spaces on the exits are allowed. For example, in the above Java code, S is a statement and T takes over in case exception e is raised in S. On the exceptional exit of S, there would be an additional object e holding the information of the exception, thus the exceptional exit state space differs from the one of normal exit. Similarly, in the above Eiffel code of method M, T handles exceptions raised in S, while P and Q are the pre- and postconditions respectively. T is executed if S raises an exception. If T terminates normally and it is followed by a **retry** statement, then M will be re-executed; if T terminates with an exception, then the whole method M terminates with an exception. On the normal exit, the local variables *vars* are excluded from the state space, while on the exceptional and retry exits they are kept. Secondly, the formalization syntactically separates postconditions of exits, making them more comprehensible than when postconditions of exits are mixed together. This methodology has been adopted successfully in our previous work for verification rules in two different exception handling mechanisms [21] and [22], including rules for loop verification. In this work we focus on a more general version, capable of dealing with arbitrary number of exits with potentially different state spaces. Isabelle/HOL was chosen as the proof assistant, due to its built-in support of higher-order logic, extensive theories, and detailed documentation.

The contribution of the present work is a collection of algebraic properties for multi-exit statements and normal forms for three classes of statements, viz. monotonic, positively conjunctive, and universally conjunctive statements. The normal form for monotonic statements allows an arbitrary statement with n exits to be equivalently expressed by $n+1$ relations, one relation for an angelic update followed by one relation for a demonic update for each exit. Positively conjunctive statements characterize abstract programs and can be taken as a domain in which to formalize program development, e.g. [1, 3, 19]. The normal form allows an arbitrary positively conjunctive statement with n exits to be equivalently expressed by a predicate, the precondition, and n relations, one for each exit. In addition, the predicate is unique. The normal form of universally conjunctive statements allows an arbitrary universally conjunctive statement with n exits to be equivalently expressed by n relations, one for each exit.

The next section develops the model of multi-exit statements as monotonic higher-order predicate transformers with indexed postconditions. Section 3 introduces the main subclasses of statements, viz. positively conjunctive, universally conjunctive, positively disjunctive, and universally disjunctive statements. Section 4 discusses the algebraic structure of multi-exit statements. Sections 5 and 6 give the normal forms for monotonic, positively conjunctive, and universally conjunctive statements. The final section discusses the formalization of the theory in Isabelle/HOL and gives an application to the design of specification languages.

2 Predicate Transformers with Indexed Postconditions

The values of type *Bool* are the truth values true and false. We write the equality of truth values $\equiv$ instead of $=$. Truth values form a lattice with conjunction $\wedge$ as the meet, disjunction $\vee$ as the join, $\Rightarrow$ as the order relation, false as the bottom element, and true as the top element. The lattice is complete, with $\forall x \bullet b x$ and $\exists x \bullet b x$ being the meet and the join of the set $\{x \mid b x\}$. The lattice is also distributive as $\wedge$ distributes over $\vee$ and $\vee$ distributes over $\wedge$. This extends to *infinite distributivity*, in the sense that $a \vee (\forall x \bullet b x) \equiv (\forall x \bullet a \vee b x)$ and $a \wedge (\exists x \bullet b x) \equiv (\exists x \bullet a \wedge b x)$.

A *state predicate* of type $\mathscr{P}\Sigma$ is a function from a state space Σ to *Bool*, i.e. a function of type $\Sigma \to Bool$. On state predicates, conjunction $\wedge$, disjunction $\vee$, implication $\Rightarrow$, negation $\neg$, universal quantification $\forall$, and existential quantification $\exists$ are defined by the pointwise extension of the corresponding operations on *Bool*. For example, for state predicates $p, q : \mathscr{P}\Sigma$ and state $\sigma : \Sigma$, we have that $(p \wedge q)\,\sigma \equiv p\,\sigma \wedge q\,\sigma$ and for $p : \alpha \to \mathscr{P}\Sigma$ we have $(\forall v : \alpha \bullet p\,v)\,\sigma \equiv (\forall v : \alpha \bullet p\,v\,\sigma)$. As we use a typed logic, we assume that types can be inferred, e.g. we write simply $(\forall v \bullet p\,v)$. Entailment $\leq$ is defined by universal implication, $p \leq q \equiv \forall \sigma \bullet p\,\sigma \Rightarrow q\,\sigma$. The predicates true and false represent the universally true and false predicates, e.g. $\mathsf{true}\,\sigma \equiv \mathsf{true}$. The pointwise extension of a (complete, distributive) lattice is again a (complete, distributive) lattice. Therefore state predicates with $\leq$ as the order relation form a complete distributive lattice.

For a tuple $t = (t_1, \ldots, t_n)$, we write $t\,i$ for selecting its i-th element. An *indexed state predicate* is a tuple $(p_1, \ldots, p_n)$ of state predicates p_i. We call $\mathscr{X} = \{1, \ldots, n\}$ its *exit indices* and write the type $\mathscr{P}\Sigma_1 \times \cdots \times \mathscr{P}\Sigma_n$ of an indexed state predicate more concisely as $\mathscr{P}\Sigma_{\mathscr{X}}$. The exit $\mathsf{nrl} = 1$ is the *normal exit*. On indexed state predicates, conjunction $\wedge$, disjunction $\vee$, implication $\Rightarrow$, negation $\neg$, universal quantification $\forall$, and existential quantification $\exists$ are defined by the elementwise extension of the corresponding operations on state predicates. For example, for indexed state predicates $P, Q : \mathscr{P}\Sigma_{\mathscr{X}}$ and exit index $x \in \mathscr{X}$, we have that $(P \wedge Q)x = (Px \wedge Qx)$ and for $P : \alpha \to \mathscr{P}\Sigma_{\mathscr{X}}$ we have that $(\forall v \bullet Pv)x = (\forall v \bullet Pvx)$. Entailment $\leq$ is defined by elementwise entailment of state predicates, $P \leq Q \equiv \wedge x \bullet Px \leq Qx$ for $P, Q : \mathscr{P}\Sigma_{\mathscr{X}}$. The indexed state predicates $\overline{\mathsf{true}}$ and $\overline{\mathsf{false}}$ represent the elementwise true and false state predicates, e.g. $\overline{\mathsf{true}}\,x = \mathsf{true}$. As indexed state predicates are an elementwise extension of state predicates, indexed state predicates with $\leq$ as the order relation form a complete, distributive lattice.

A *predicate transformer (with multiple exits)* is a function from an indexed state predicate, the postcondition for each exit, to a state predicate, the precondition, i.e. is a function of type $\mathscr{P}\Gamma_{\mathscr{X}} \to \mathscr{P}\Sigma$.

We define some basic predicate transformers: abort may terminate at any exit in any state or may not terminate at all; stop satisfies miraculously any postcondition on any exit by blocking execution; skip_x does not change the state and terminates at exit x. With indexed state predicate Q we define:

$$\begin{aligned} \mathsf{abort}\, Q &\;\widehat{=}\; \mathsf{false} \\ \mathsf{stop}\, Q &\;\widehat{=}\; \mathsf{true} \\ \mathsf{skip}_x\, Q &\;\widehat{=}\; Qx \end{aligned}$$

Let S, T be predicate transformers. The *sequential composition* $S \,;_x T$ executes first S and provided that S terminates at exit x, continues with T. The *demonic choice* $S \sqcap T$ establishes a postcondition at an exit if both S and T do. The *angelic choice* $S \sqcup T$ establishes a postcondition at an exit if either S or T does. We write $t[i \leftarrow v]$ for modifying tuple t to be v at index i:

$$\begin{aligned} (S\,;_x T)\, Q &\;\widehat{=}\; S(Q[x \leftarrow T\,Q]) \\ (S \sqcap T)\, Q &\;\widehat{=}\; S\,Q \wedge T\,Q \\ (S \sqcup T)\, Q &\;\widehat{=}\; S\,Q \vee T\,Q \end{aligned}$$

Binary choice generalizes to choice over arbitrary sets. Let $\mathscr{S}$ be a set of predicate transformers:

$$\begin{aligned} (\sqcap \mathscr{S})\, Q &\;\widehat{=}\; (\forall S \in \mathscr{S} \bullet S\,Q) \\ (\sqcup \mathscr{S})\, Q &\;\widehat{=}\; (\exists S \in \mathscr{S} \bullet S\,Q) \end{aligned}$$

The demonic choice over the empty set blocks, $\sqcap\emptyset = \mathsf{stop}$, and the angelic choice over the empty set aborts, $\sqcup\emptyset = \mathsf{abort}$. We write $\sqcap v \bullet S\,v$ for $\sqcap\{v \mid S\,v\}$ and $\sqcup v \bullet S\,v$ for $\sqcup\{v \mid S\,v\}$.

Let P be an indexed state predicate. The *assumption* $[P]$ allows continuation at exit x if $Px = \mathsf{true}$. If continuation at several exits is possible, the choice is demonic. If continuation is not possible, the assumption stops. The *assertion* $\{P\}$ also allows continuation at exit x if $Px = \mathsf{true}$. If continuation at several exits is possible, the choice is angelic. If continuation is not possible, the assumption aborts:

$$\begin{aligned} [P]\, Q &\;\widehat{=}\; \wedge x \bullet Px \Rightarrow Qx \\ \{P\}\, Q &\;\widehat{=}\; \vee x \bullet Px \wedge Qx \end{aligned}$$

We have that $[\overline{\mathsf{false}}] = \mathsf{stop}$ and that $[\overline{\mathsf{true}}] = \sqcap x \bullet \mathsf{skip}_x$. Dually, we have that $\{\overline{\mathsf{false}}\} = \mathsf{abort}$ and that $\{\overline{\mathsf{true}}\} = \sqcup x \bullet \mathsf{skip}_x$. For state predicates $p_1, \ldots, p_k$ and distinct $x_1, \ldots, x_k \in \mathscr{X}$ we write $x_1 \mapsto p_1, \ldots, x_k \mapsto p_k$ for the indexed state predicate that is p_1 at x_1, ..., p_k at x_k, and false everywhere else. As a special case, $x \mapsto \mathsf{true}$ is the indexed state predicate that is true for $x \in \mathscr{X}$ and false otherwise. In the case of assertions and assumptions with only one predicate being true and all other false, the continuation is deterministic, in the sense that $[x \mapsto \mathsf{true}] = \mathsf{skip}_x = \{x \mapsto \mathsf{true}\}$. If only a predicate for the normal exit is specified, we write $[p]$ for $[\mathsf{nrl} \mapsto p]$ and likewise $\{p\}$ for $\{\mathsf{nrl} \mapsto p\}$.

Theorem 1 *Let p be a state predicate and Q be an indexed state predicate:*

$$\begin{aligned} [p]\,Q &= (p \Rightarrow Q\,\mathsf{nrl}) \\ \{p\}\,Q &= (p \wedge Q\,\mathsf{nrl}) \end{aligned}$$

A *(state) relation* is a function of type $\Sigma \to \mathscr{P}\Gamma$. The *empty relation* $\bot$ is defined by $\bot\,\sigma\,\gamma \equiv \mathsf{false}$, the *universal relation* $\top$ is defined by $\top\,\sigma\,\gamma \equiv \mathsf{true}$, and the *identity relation* id by $\mathsf{id}\,\sigma\,\sigma' \equiv \sigma = \sigma'$. Intersection $\cap$, union $\cup$, complement $\bar{r}$, and composition $\circ$ can be defined straightforwardly, but are not needed here; *inclusion* is defined by $r \subseteq r' \equiv \forall \sigma \bullet r\,\sigma \leq r'\,\sigma$. An *indexed (state) relation* is a tuple of relations with identical initial state space but possibly different final state spaces, i.e. is of type $(\Sigma \to \mathscr{P}\Gamma_1) \times \cdots \times (\Sigma \to \mathscr{P}\Gamma_n)$. With $\mathscr{X} = \{1, \ldots, n\}$ we write this more concisely as $\Sigma \overset{\mathscr{X}}{\to} \mathscr{P}\Gamma$. We write $\overline{\top}$, $\overline{\bot}$, and $\overline{\mathsf{id}}$ for the indexed relations that are universally $\top$, $\bot$, and id. For relations $r_1, \ldots, r_k$ and $x_1, \ldots, x_k \in \mathscr{X}$ we write $R = x_1 \mapsto r_1, \ldots, x_k \mapsto r_k$ for the indexed relation R that is r_1 at x_1, ..., r_n at x_n, and $\bot$ everywhere else.

An *update* specifies a state change by one relation for each exit, each relation relating the common initial state to the final state at each exit. Let R be an indexed relation. The *demonic update* $[R]$ allows continuation at exit x from initial state σ if $R\,x\,\sigma$ specifies some final states. The choice among all possible final states and the choice among all possible exits are demonic. If continuation is not possible, the demonic update stops. Dually, the *angelic update* $\{R\}$ allows continuation at exit x from initial state σ if $R\,x\,\sigma$ specifies some final states. The choice among all possible final states and the choice among all possible exits are angelic. If continuation is not possible, the angelic update aborts:

$$\begin{aligned} [R]\,Q\,\sigma &\mathrel{\widehat{=}} \wedge x \bullet \forall \gamma \bullet R\,x\,\sigma\,\gamma \Rightarrow Q\,x\,\gamma \\ \{R\}\,Q\,\sigma &\mathrel{\widehat{=}} \vee x \bullet \exists \gamma \bullet R\,x\,\sigma\,\gamma \wedge Q\,x\,\gamma \end{aligned}$$

We have that $[\overline{\bot}] = \mathsf{stop}$ and that $[\overline{\mathsf{id}}] = \sqcap x \bullet \mathsf{skip}_x$. Dually, we have that $\{\overline{\bot}\} = \mathsf{abort}$ and that $\{\overline{\mathsf{id}}\} = \sqcup x \bullet \mathsf{skip}_x$. Both updates $[\overline{\top}]$ and $\{\overline{\top}\}$ always terminate in some state at some exit, with $[\overline{\top}]$ making the choice among the exits and among the states demonic and $\{\overline{\top}\}$ making these choices angelic. In the case of updates with only one relation being the identity and all other relations being empty, the continuation is deterministic, in the sense that $[x \mapsto \mathsf{id}] = \mathsf{skip}_x = \{x \mapsto \mathsf{id}\}$. For the case that only a relation for the normal exit is specified, we write $[r]$ for $[\mathsf{nrl} \mapsto r]$ and likewise $\{r\}$ for $\{\mathsf{nrl} \mapsto r\}$.

Theorem 2 *Let r be a relation and Q be an indexed state predicate:*

$$\begin{aligned} [r]\,Q\,\sigma &= \forall \gamma \bullet r\,\sigma\,\gamma \Rightarrow Q\,\mathsf{nrl}\,\gamma \\ \{r\}\,Q\,\sigma &= \exists \gamma \bullet r\,\sigma\,\gamma \wedge Q\,\mathsf{nrl}\,\gamma \end{aligned}$$

For an indexed state predicate $P : \mathscr{P}\Sigma_{\mathscr{X}}$, the *lifting* $|P| : \Sigma \overset{\mathscr{X}}{\to} \mathscr{P}\Sigma$ is an indexed state relation that, for each exit, is a partial identity relation:

$$|P|\,x\,\sigma\,\sigma' \mathrel{\widehat{=}} P\,x\,\sigma \wedge \sigma = \sigma'$$

Assumptions and assertions can be expressed as demonic and angelic updates by lifting their argument to an indexed state relation:

Theorem 3 *Let P be an indexed state predicate:*

$$\begin{array}{rcl} [P] & = & [|P|] \\ \{P\} & = & \{|P|\} \end{array}$$

A *multi-exit statement* S is a predicate transformer that is constructed of

- *basic statements* abort, stop, skip_x, $[P]$, $\{P\}$, $[R]$, $\{R\}$, where x is an exit index, P is an indexed state predicate, R is an indexed relation, and
- *composed statements* $S_1 \,;_x S_2$, $\sqcap \mathscr{S}$, $\sqcup \mathscr{S}$, where S_1, S_2 are statements and $\mathscr{S}$ is a set of statements.

The *normal* skip statement and the *normal sequential composition* are special cases:

$$\begin{array}{rcl} \mathsf{skip} & = & \mathsf{skip}_{\mathsf{nrl}} \\ S_1 \,; S_2 & = & S_1 \,;_{\mathsf{nrl}} S_2 \end{array}$$

Exception handling is expressed in terms of the statement $\mathsf{raise}\, x$ to raise exception x and the statement $\mathsf{try}\, S_1\, \mathsf{on}\, x\, \mathsf{do}\, S_2$ to start with S_1 and on exception x to continue with S_2, otherwise to continue normally. With exceptions being exits, these are defined as:

$$\begin{array}{rcll} \mathsf{raise}\, x & = & \mathsf{skip}_x & \quad \text{if } x \neq \mathsf{nrl} \\ \mathsf{try}\, S_1\, \mathsf{on}\, x\, \mathsf{do}\, S_2 & = & S_1 \,;_x S_2 & \quad \text{if } x \neq \mathsf{nrl} \end{array}$$

3 Monotonicity, Junctivity, and Domains

We characterize different classes of statements. Predicate transformer S is *monotonic* if $P \leq Q \Rightarrow S\,P \leq S\,Q$ for arbitrary indexed state predicates P, Q. An immediate consequence is:

Theorem 4 *All basic statements are monotonic and all composed statements preserve monotonicity.*

Positive "junctivity" properties are defined by distributivity over arbitrary, but non-empty sets of indexed state predicates. Predicate transformer S is *positively conjunctive* if $S\,(\forall Q \in \mathscr{Q} \bullet Q) = (\forall Q \in \mathscr{Q} \bullet S\,Q)$ and *positively disjunctive* if $S\,(\exists Q \in \mathscr{Q} \bullet Q) = (\exists Q \in \mathscr{Q} \bullet S\,Q)$ for arbitrary set $\mathscr{Q} \neq \emptyset$ of indexed state predicates.

Theorem 5 *Statements* stop, skip_x*, assertion* $\{P\}$*, assumption* $[P]$*, and demonic update* $[R]$ *are positively conjunctive. Sequential composition* $;_x$ *and demonic choice* $\sqcap$ *preserve positive conjunctivity.*

Proof We give the proof only for the demonic update $[R]$. Let $\mathscr{Q}$ be a set of indexed predicates and σ a state:

$$\begin{array}{ll} & [R]\,(\forall Q \in \mathscr{Q} \bullet Q)\,\sigma \\ \equiv & \quad \langle \text{definition of demonic update} \rangle \\ & \wedge x \bullet \forall \gamma \bullet R\,x\,\sigma\,\gamma \Rightarrow (\forall Q \in \mathscr{Q} \bullet Q)\,x\,\gamma \\ \equiv & \quad \langle \forall \text{ infinite distributivity, } \forall \text{ commutativity} \rangle \\ & \forall Q \in \mathscr{Q} \bullet \wedge x \bullet \forall \gamma \bullet R\,x\,\sigma\,\gamma \Rightarrow Q\,x\,\gamma \\ \equiv & \quad \langle \text{definition of demonic update} \rangle \\ & \forall Q \in \mathscr{Q} \bullet [R]\,Q\,\sigma \\ \equiv & \quad \langle \text{definition of } \forall \text{ for state predicates} \rangle \\ & (\forall Q \in \mathscr{Q} \bullet [R]\,Q)\,\sigma \end{array}$$

Thus $[R]$ is positively conjunctive. ■

Angelic choice does not preserve positive conjunctivity in general. A dual theorem holds for positively disjunctive statements.

Theorem 6 *Statements* abort *and* skip_x*, assumption* $[P]$*, assertion* $\{P\}$*, and angelic update* $\{R\}$ *are positively disjunctive. Sequential composition* $;_x$ *and angelic choice* $\sqcup$ *preserve positive disjunctivity.*

Universal "junctivity" properties are defined by distributivity over arbitrary sets of indexed state predicates. Predicate transformer S is *universally conjunctive* if $S\,(\forall\,Q \in \mathscr{Q} \bullet Q) = (\forall\,Q \in \mathscr{Q} \bullet S\,Q)$ and *universally disjunctive* if $S\,(\exists\,Q \in \mathscr{Q} \bullet Q) = (\exists\,Q \in \mathscr{Q} \bullet S\,Q)$ for arbitrary set $\mathscr{Q}$ of indexed state predicates.

Theorem 7 *Any universally conjunctive predicate transformer is positively conjunctive. Any positively conjunctive predicate transformer is monotonic.*

Proof The first part follows immediately from the definitions. For positively conjunctive predicate transformer S, we show that $S\,P \le S\,Q$ provided $P \le Q$:

$$\begin{array}{ll} & S\,P \\ = & \quad \langle \text{assumption } P \le Q \rangle \\ & S\,(P \wedge Q) \\ = & \quad \langle \text{assumption } S \text{ is positively conjunctive} \rangle \\ & S\,P \wedge S\,Q \\ \le & \quad \langle \text{property of } \le \rangle \\ & S\,Q \end{array}$$

Thus S is monotonic. ■

It is easy to see that abort is not universally conjunctive by taking $\mathscr{Q} = \emptyset$. As a consequence, an assertion $\{P\}$ is in general not universally conjunctive either. Angelic choice does not preserve universal conjunctivity. The relationship between universal and positive "junctivity" can be stated more precisely by considering domains. For predicate transformer S, the *termination domain* and *enabledness domain* are defined by:

$$\begin{array}{rcl} \mathsf{tr}\,S & = & S\,\overline{\mathsf{true}} \\ \mathsf{en}\,S & = & \neg\,(S\,\overline{\mathsf{false}}) \end{array}$$

We say that S is *terminating* if $\mathsf{tr}\,S = \mathsf{true}$ and that S is *enabled* (or *strict*) if $\mathsf{en}\,S = \mathsf{true}$.

Theorem 8 *A predicate transformer is universally conjunctive if and only if it is positively conjunctive and terminating.*

As an immediate consequence, stop, skip_x, assumption $[P]$, and demonic update $[R]$ are universally conjunctive. Also, sequential composition $;_x$ and demonic choice $\sqcap$ preserve universal conjunctivity. Dually, we have:

Theorem 9 *A predicate transformer is universally disjunctive if and only if it is positively disjunctive and enabled.*

The termination domain $\mathsf{tr}_x\, S$ characterizes those initial states from which termination at exit x is guaranteed. More generally, for a set $X \subseteq \mathscr{X}$ the state predicate $\mathsf{tr}_X\, S$ characterizes those initial states from which termination at any of the exits of X is guaranteed. Writing $X \mapsto p$ for the indexed state predicate that is p at all $x \in X$ and false otherwise, we define:

$$\mathsf{tr}_X\, S \quad = \quad S(X \mapsto \mathsf{true})$$

By monotonicity we have that $\mathsf{tr}_X\, S \leq \mathsf{tr}\, S$ and that $\mathsf{tr}_X\, S \vee \mathsf{tr}_Y\, S \leq \mathsf{tr}_{X \cup Y}\, S$. To see that equality does not always hold, consider $X = \{x, y\}$. For $S = \mathsf{skip}_x \sqcap \mathsf{skip}_y$ we have that $\mathsf{false} = \mathsf{tr}_{\{x\}}\, S < \mathsf{tr}\, S = \mathsf{true}$. We also have that $\mathsf{false} = \mathsf{tr}_{\{x\}}\, S \vee \mathsf{tr}_{\{y\}}\, S < \mathsf{tr}_{\{x,y\}}\, S = \mathsf{true}$.

4 Algebraic Properties

For fixed exit index x, indexed predicate transformers with $;_x$ as composition and skip_x as unit form a monoid. For distinct exit indices x, y, we only have that skip_x is a left zero of $;_y$.

Theorem 10 *Let $\mathscr{X}$ be the exit indices and let S, T, U be indexed predicate transformers. For any $x, y \in \mathscr{X}$ we have:*

$$\begin{aligned}
(S ;_x T) ;_x U &= S ;_x (T ;_x U) && \text{(a)}\\
S ;_x \mathsf{skip}_x &= S && \text{(b)}\\
\mathsf{skip}_x ;_x S &= S && \text{(c)}\\
\mathsf{skip}_x ;_y S &= \mathsf{skip}_x \quad \textit{if} \quad x \neq y && \text{(d)}
\end{aligned}$$

Proof For (a), we calculate for any indexed predicate Q:

$$\begin{aligned}
& ((S ;_x T) ;_x U)\, Q\\
= \quad & \quad \langle \text{definition of } ;_x \rangle\\
& (S ;_x T)\, (Q[x \leftarrow U\, Q])\\
= \quad & \quad \langle \text{definition of } ;_x \rangle\\
& S\, (Q[x \leftarrow U\, Q]\, [x \leftarrow T\, (Q[x \leftarrow U\, Q])])\\
= \quad & \quad \langle \text{for any } a, b, c, t\text{: } t[a \leftarrow b][a \leftarrow c] = t[a \leftarrow c] \rangle\\
& S\, (Q[x \leftarrow T\, (Q[x \leftarrow U\, Q])])\\
= \quad & \quad \langle \text{definition of } ;_x \rangle\\
& S\, (Q[x \leftarrow (T ;_x U)\, Q])\\
= \quad & \quad \langle \text{definition of } ;_x \rangle\\
& (S ;_x (T ;_x U))\, Q
\end{aligned}$$

For (b) we have:

$$\begin{aligned}
& (S ;_x \mathsf{skip}_x)\, Q\\
= \quad & \quad \langle \text{definition of } ;_x \rangle\\
& S\, (Q[x \leftarrow \mathsf{skip}_x\, Q])\\
= \quad & \quad \langle \text{definition of } \mathsf{skip}_x \rangle\\
& S\, (Q[x \leftarrow Q x])\\
= \quad & \quad \langle \text{for any } a, t\text{: } t[a \leftarrow t\, a] = t \rangle\\
& S\, Q
\end{aligned}$$

For (c) we have:

$$\begin{array}{ll}
& (\mathsf{skip}_x \,;_x S)\, Q \\
= & \quad \langle \text{definition of } ;_x \rangle \\
& \mathsf{skip}_x (Q[x \leftarrow S\,Q]) \\
= & \quad \langle \text{definition of } \mathsf{skip}_x \rangle \\
& (Q[x \leftarrow S\,Q])\, x \\
= & \quad \langle \text{for any } a,b,t\text{: } (t[a \leftarrow b])\, a = b \rangle \\
& S\,Q
\end{array}$$

Assuming $x \neq y$, we have for (d):

$$\begin{array}{ll}
& (\mathsf{skip}_x \,;_y S)\, Q \\
= & \quad \langle \text{definition of } ;_y \rangle \\
& \mathsf{skip}_x (Q[y \leftarrow S\,Q]) \\
= & \quad \langle \text{definition of } \mathsf{skip}_x \rangle \\
& (Q[y \leftarrow S\,Q])\, x \\
= & \quad \langle \text{for any } a,b,c,t \text{ with } a \neq c\text{: } (t[a \leftarrow b])\, c = t\, c \rangle \\
& \mathsf{skip}_x\, Q
\end{array}$$

As the unit of a monoid is unique, we have that skip_x is the unique identity of $;_x$. To see that $(S\,;_x T)\,;_y U \neq S\,;_x (T\,;_y U)$ in general, consider that $x = \mathsf{nrl}$ and $y \neq \mathsf{nrl}$. Rewriting using try statements, it is intuitive that $\mathsf{try}\, S\,; T\, \mathsf{on}\, z\, \mathsf{do}\, U$ is in general different from $S\,; \mathsf{try}\, T\, \mathsf{on}\, y\, \mathsf{do}\, U$. Assuming $x, y \neq \mathsf{nrl}$ and rewriting $S\,;_y \mathsf{skip}_x$ as $\mathsf{try}\, S\, \mathsf{on}\, y\, \mathsf{do}\, \mathsf{raise}\, x$ we obtain the idiom of *re-raising* an exception; in this case, skip_x (or $\mathsf{raise}\, x$) is neither unit nor zero.

The *refinement* relation is defined by universal entailment of indexed state predicates. For indexed predicate transformers S, T we define:

$$S \sqsubseteq T \quad \mathrel{\widehat{=}} \quad \forall Q \bullet S\,Q \leq T\,Q$$

Intuitively, refinement may reduce demonic choice and may increase angelic choice, where the choice is between exits or states. For example, $\mathsf{skip}_x \sqcap \mathsf{skip}_y \sqsubseteq \mathsf{skip}_x$. This is captured by the lattice structure of indexed predicate transformers.

Theorem 11 *Indexed predicate transformers with $\sqsubseteq$ as the order relation,* abort *as bottom,* stop *as top, $\sqcap$ as meet, and $\sqcup$ as join form a complete distributive lattice.*

This follows immediately from indexed predicate transformers being a pointwise extension of state predicates. The monoid and lattice structure are connected by following distributivity properties.

Theorem 12 *Let S, T be indexed predicate transformers and $\mathscr{S}, \mathscr{T}$ be non-empty sets of indexed predicate transformers such that $\mathscr{X}$ is the index set of S and all elements of $\mathscr{S}$:*

$$(\sqcap S \in \mathscr{S} \bullet S)\,;_x T = (\sqcap S \in \mathscr{S} \bullet S\,;_x T) \qquad \text{(a)}$$

$$S\,;_x (\sqcap T \in \mathscr{T} \bullet T) \sqsubseteq (\sqcap T \in \mathscr{T} \bullet S\,;_x T) \quad \textit{if } S \textit{ is monotonic} \qquad \text{(b)}$$

$$S\,;_x (\sqcap T \in \mathscr{T} \bullet T) = (\sqcap T \in \mathscr{T} \bullet S\,;_x T) \quad \textit{if } S \textit{ is positively conjunctive} \qquad \text{(c)}$$

Proof For (a), we calculate for any indexed predicate Q:

$$
\begin{array}{ll}
& ((\sqcap S \in \mathscr{S} \bullet S) \,;_x T)\, Q \\
= & \quad \langle \text{definition of sequential composition} \rangle \\
& (\sqcap S \in \mathscr{S} \bullet S)\, (Q[x \leftarrow T\, Q]) \\
= & \quad \langle \text{definition of demonic choice} \rangle \\
& (\forall S \in \mathscr{S} \bullet S\, (Q[x \leftarrow T\, Q])) \\
= & \quad \langle \text{definition of sequential composition} \rangle \\
& (\forall S \in \mathscr{S} \bullet (S \,;_x T)\, Q) \\
= & \quad \langle \text{definition of demonic choice} \rangle \\
& (\sqcap S \in \mathscr{S} \bullet S \,;_x T)\, Q
\end{array}
$$

For (b) we have:

$$
\begin{array}{ll}
& (S \,;_x (\sqcap T \in \mathscr{T} \bullet T))\, Q \\
= & \quad \langle \text{definition of sequential composition} \rangle \\
& S(Q[x \leftarrow (\sqcap T \in \mathscr{T} \bullet T)\, Q]) \\
= & \quad \langle \text{definition of demonic choice} \rangle \\
& S(Q[x \leftarrow (\forall T \in \mathscr{T} \bullet T\, Q)]) \\
\Leftarrow & \quad \langle \text{monotonicity (*)} \rangle \\
& \forall T \in \mathscr{T} \bullet S\, (Q[x \leftarrow T\, Q]) \\
= & \quad \langle \text{definition of sequential composition} \rangle \\
& \forall T \in \mathscr{T} \bullet (S \,;_x T)\, Q \\
= & \quad \langle \text{definition of demonic choice} \rangle \\
& (\sqcap T \in \mathscr{T} \bullet S \,;_x T)\, Q
\end{array}
$$

The proof of (c) is similar, except that it uses positive conjunctivity of S in step (*).

5 Normal Form for Monotonic Predicate Transformers

Every multi-exit statement is a monotonic indexed predicate transformer by definition. We show that the converse also holds, that every monotonic indexed predicate transformer can be equivalently expressed as a multi-exit statement. It turns out that only normal angelic update, normal sequential composition, and demonic update are needed to express any monotonic indexed predicate transformer. Following theorem and its proof generalize those of [3] for single-exit statements.

Theorem 13 *Let S be a monotonic indexed predicate transformer. Then there exist a relation r and an indexed relation R such that $S = \{r\} \,;[R]$.*

Proof Assume that $S : \mathscr{P}\Gamma_{\mathscr{X}} \to \mathscr{P}\Sigma$ is a monotonic indexed predicate transformer. Define relation $r : \Sigma \to \mathscr{P}(\mathscr{P}\Gamma_{\mathscr{X}})$ and indexed relation $R : \mathscr{P}\Gamma_{\mathscr{X}} \overset{\mathscr{X}}{\to} \mathscr{P}\Gamma$ as follows, where P is an indexed state predicate of type $\mathscr{P}\Gamma_{\mathscr{X}}$:

$$
\begin{array}{rcl}
r\sigma P & \widehat{=} & SP\sigma \\
RxP\gamma & \widehat{=} & Px\gamma
\end{array}
$$

Then, for arbitrary indexed predicate Q and arbitrary state σ_0 we have:

$$\begin{array}{ll}
 & (\{r\}\,;[R])\,Q\,\sigma_0 \\
\equiv & \quad \langle \text{definition of sequential composition} \rangle \\
 & \{r\}\,(Q[\mathsf{nrl} \leftarrow [R]\,Q])\,\sigma_0 \\
\equiv & \quad \langle \text{definition of demonic update} \rangle \\
 & \{r\}\,(Q[\mathsf{nrl} \leftarrow (\lambda\,\sigma \bullet \wedge x \bullet \forall \gamma \bullet R\,x\,\sigma\,\gamma \Rightarrow Q\,x\,\gamma)])\,\sigma_0 \\
\equiv & \quad \langle \text{Theorem 2, definition of } R \rangle \\
 & (\exists P \bullet r\,\sigma_0\,P \wedge (\wedge x \bullet \forall \gamma \bullet P\,x\,\gamma \Rightarrow Q\,x\,\gamma)) \\
\equiv & \quad \langle \text{definition of } r\text{, definition of } \leq \rangle \\
 & (\exists P \bullet S\,P\,\sigma_0 \wedge P \leq Q) \\
\equiv & \quad \langle (^{**}) \rangle \\
 & S\,Q\,\sigma_0
\end{array}$$

The step (**) is shown by mutual implication:

$$\begin{array}{ll}
 & (\exists P \bullet S\,P\,\sigma_0 \wedge P \leq Q) \\
\Rightarrow & \quad \langle S \text{ is monotonic, context says } P \leq Q \rangle \\
 & (\exists P \bullet S\,Q\,\sigma_0 \wedge P \leq Q) \\
\Rightarrow & \quad \langle \text{weakening} \rangle \\
 & (\exists P \bullet S\,Q\,\sigma_0) \\
\Rightarrow & \quad \langle \text{quantifier rule} \rangle \\
 & S\,Q\,\sigma_0
\end{array}$$

For the reverse implication we have:

$$\begin{array}{ll}
 & (\exists P \bullet S\,P\,\sigma_0 \wedge P \leq Q) \\
\Leftarrow & \quad \langle \text{witness } P = Q \rangle \\
 & S\,Q\,\sigma_0 \wedge Q \leq Q \\
\equiv & \quad \langle \text{reflexitivity} \rangle \\
 & S\,Q\,\sigma_0
\end{array}$$

Together this shows that $S = \{r\}\,;[R]$. ■

As a consequence, any monotonic indexed predicate transformer can be expressed as a statement. The normal form $\{r\}\,;[R]$ can be interpreted as a simple two-player game [3]: given an initial state, first the angel chooses an intermediate exit and an intermediate state. Then, for any possible intermediate exit and intermediate state, the demon choses the final exit and final state. An interesting observation is how the intermediate state space is constructed in the proof. The precondition is of type $\mathscr{P}\Sigma$, the indexed postcondition is of type $\mathscr{P}\Gamma_{\mathscr{X}}$, and the intermediate condition is of type $\mathscr{P}(\mathscr{P}\Gamma_{\mathscr{X}})$. That is, an intermediate state is a set of final states. Given initial state σ, the angel first picks an indexed state predicate P of type $\mathscr{P}\Gamma_{\mathscr{X}}$ such that $S\,P\,\sigma$ holds and then the demon picks at each exit x a final state γ such that $P\,x\,\gamma$ holds.

To illustrate this with a concrete example, consider statement S with two exists, a normal and an exceptional exit, $\mathscr{X} = \{\mathsf{nrl}, \mathsf{exc}\}$. Recalling that $\mathsf{skip} = \mathsf{skip}_{\mathsf{nrl}}$, we define $\mathsf{raise} = \mathsf{skip}_{\mathsf{exc}}$ and $S = \mathsf{skip} \sqcup \mathsf{raise}$. The weakest precondition of S is $S\,(q_1,q_2) = (q_1 \vee q_2)$. The normal form of S is $\{r\};[R]$ such that $r\,p\,(q_1,q_2) \equiv p = (q_1 \vee q_2)$ and $R = (r_1,r_2)$ where $r_1\,(q_1,q_2)\,\gamma_1 \equiv q_1\,\gamma_1$ and $r_2\,(q_1,q_2)\,\gamma_2 \equiv q_2\,\gamma_2$. From any precondition p, the normal angelic update $\{r\} = \{(r,\bot)\}$ always exits normally. However, intuitively it can choose to go to the intermediate state (p, false), which represents either exiting normally with postcondition p or exiting exceptionally with false, it can also choose to go to the intermediate state (false, p), which represents either exiting normally with postcondition false or exiting exceptionally with p, or it can choose to establish any (q_1,q_2) where $p = (q_1 \vee q_2)$. In case of (p, false), the next statement, $[(r_1,r_2)]$, forces r_1 to be chosen (r_2 is $\bot$ due to false, according to the definition) and S exits normally with p, as skip does. In case of (false, p), the next statement, $[(r_1,r_2)]$, forces r_2 to be chosen (r_1 is $\bot$ due to false, according to the definition) and S exits exceptionally with p, as raise does.

As a second example, consider statement S defined by $S = \mathsf{skip} \sqcap \mathsf{raise}$. The weakest precondition of S is $S(q_1,q_2) = (q_1 \wedge q_2)$. The normal form of S is $\{r\};[R]$ such that $r\,p\,(q_1,q_2) \equiv p = (q_1 \wedge q_2)$, and $R = (r_1,r_2)$ where $r_1\,(q_1,q_2)\,\gamma_1 \equiv q_1\,\gamma_1$, $r_2\,(q_1,q_2)\,\gamma_2 \equiv q_2\,\gamma_2$. From any precondition p, the normal angelic update $\{r\} = \{(r,\bot)\}$ exits normally. It can choose to go in the state (p,p) or it can choose to establish and (q_1,q_2) where $p = (q_1 \wedge q_2)$. In case of (p,p) in the intermediate state, the next statement, $[(r_1,r_2)]$ can choose r_1 and exit normally with p, as skip does, or choose r_2 and exit exceptionally with p, as raise does.

A further observation is that, in principle, sequential composition at exit other than nrl, general angelic updates, demonic choice, and angelic choice are not needed to express an arbitrary monotonic predicate transformer. Still, we consider them for symmetry reasons and to be able to define concrete programming constructs in terms of those.

6 Normal Form for Conjunctive Predicate Transformers

Statements stop, skip_x, assumption $[P]$, and demonic update $[R]$ are universally conjunctive and sequential composition $;_x$ and demonic choice $\sqcap$ preserve universally conjunctivity. We show that in turn every universally conjunctive predicate transformer can be expressed by a statement consisting only of single demonic update, with one relation for each exit. For this, it is convenient to use *bounded quantification*. The bounded universal quantification on state predicates is defined by $(\forall v \mid b\,v \bullet p\,v)\,\sigma \equiv (\forall v \mid b\,v \Rightarrow p\,v\,\sigma)$, where $b\,v : Bool$ and $p\,v : \mathscr{P}\Sigma$. The bounded quantification on indexed state predicates is defined by $(\forall v \mid b\,v \bullet P\,v)\,x = (\forall v \mid b\,v \Rightarrow P\,v\,x)$ where $b\,v : Bool$ and $P\,v : \mathscr{P}\Sigma_{\mathscr{X}}$.

Lemma 14 *If S is a universally conjunctive predicate transformer, then for all indexed state predicates Q and all states σ:*

$$S\,Q\,\sigma \quad \equiv \quad (\forall P \mid S\,P\,\sigma \bullet P) \le Q$$

Proof When $S\,Q\,\sigma = \mathsf{true}$:

$$\begin{array}{ll} & (\forall P \mid S\,P\,\sigma \bullet P) \le Q \\ \equiv & \quad \langle \text{splitting } \forall \rangle \\ & Q \wedge (\forall P \mid S\,P\,\sigma \wedge P \ne Q \bullet P) \le Q \\ \equiv & \quad \langle \wedge\text{-elimination} \rangle \\ & \mathsf{true} \end{array}$$

When $S\,Q\,\sigma = \mathsf{false}$:

$$\begin{array}{ll}
 & (\forall P \mid S\,P\,\sigma \bullet P) \leq Q \\
\Rightarrow & \quad \langle S \text{ is monotonic, Theorem 7} \rangle \\
 & S\,(\forall P \mid S\,P\,\sigma \bullet P) \leq S\,Q \\
\equiv & \quad \langle \text{definition of conjunctivity} \rangle \\
 & (\forall P \mid S\,P\,\sigma \bullet S\,P) \leq S\,Q \\
\Rightarrow & \quad \langle \text{definition of } \leq \rangle \\
 & (\forall P \mid S\,P\,\sigma \bullet S\,P)\,\sigma \Rightarrow S\,Q\,\sigma \\
\equiv & \quad \langle \text{definition of } \forall \rangle \\
 & (\forall P \mid S\,P\,\sigma \bullet S\,P\,\sigma) \Rightarrow S\,Q\,\sigma \\
\equiv & \quad \langle \text{definition of } \forall \text{, assumption } S\,Q\,\sigma = \mathsf{false} \rangle \\
 & \mathsf{true} \Rightarrow \mathsf{false} \\
\equiv & \quad \langle \text{lattice property} \rangle \\
 & \mathsf{false}
\end{array}$$

Thus we know that $S\,Q\,\sigma \equiv (\forall P \mid S\,P\,\sigma \bullet P) \leq Q$ ■

Theorem 15 *Let S be an universally conjunctive indexed predicate transformer. There exists a unique indexed relation R such that* $S = [R]$.

Proof Let $\mathscr{X}$ be the exit indices of S. For arbitrary $x \in \mathscr{X}$ we define:

$$R\,x\,\sigma \quad = \quad (\forall P \mid S\,P\,\sigma \bullet P\,x)$$

Then we calculate for any indexed state predicate Q:

$$\begin{array}{ll}
 & [R]\,Q\,\sigma \\
\equiv & \quad \langle \text{definition of } R \text{, definition of demonic update} \rangle \\
 & \wedge x \in \mathscr{X} \bullet \forall \gamma \bullet (\forall P \mid S\,P\,\sigma \bullet P\,x)\,\gamma \Rightarrow Q\,x\,\gamma \\
\equiv & \quad \langle \text{definition of } \leq \text{ on state predicates} \rangle \\
 & \wedge x \in \mathscr{X} \bullet (\forall P \mid S\,P\,\sigma \bullet P\,x) \leq Q\,x \\
\equiv & \quad \langle \text{definition of } \leq \text{ on indexed state predicates} \rangle \\
 & (\forall P \mid S\,P\,\sigma \bullet P) \leq Q \\
\equiv & \quad \langle \text{Lemma 14} \rangle \\
 & S\,Q\,\sigma
\end{array}$$

This shows that $[R] = S$. For uniqueness we have:

$$\begin{array}{ll}
 & [R] = [R'] \\
\equiv & \quad \langle \text{antisymmetry} \rangle \\
 & [R] \sqsubseteq [R'] \wedge [R] \sqsupseteq [R'] \\
\equiv & \quad \langle \text{definition of demonic update, definition of } \leq \text{, definition of } \subseteq \rangle \\
 & \wedge x \in \mathscr{X} \bullet Rx \supseteq R'x \wedge Rx \subseteq R'x \\
\equiv & \quad \langle \text{antisymmetry} \rangle \\
 & \wedge x \in \mathscr{X} \bullet Rx = R'x \\
\equiv & \quad \langle \text{equality of tuples} \rangle \\
 & R = R'
\end{array}$$

■

Lemma 16 *If S is conjunctive, then* $[\mathsf{tr}\ S]\,;S$ *is universally conjunctive.*

Proof For arbitrary predicate S, we show that $[\mathsf{tr}\ S]\,;S$ is terminating:

$$\begin{array}{ll}
 & \mathsf{tr}\,([\mathsf{tr}\ S]\,;S) \\
= & \quad \langle \text{definition of } \mathsf{tr} \rangle \\
 & ([S\,\overline{\mathsf{true}}]\,;S)\,\overline{\mathsf{true}} \\
= & \quad \langle \text{definition of sequential composition} \rangle s \\
 & [S\,\overline{\mathsf{true}}]\,(\overline{\mathsf{true}}\,[\mathsf{nrl} \leftarrow S\,\overline{\mathsf{true}}]) \\
= & \quad \langle \text{Theorem 1} \rangle \\
 & S\,\overline{\mathsf{true}} \Rightarrow (\overline{\mathsf{true}}\,[\mathsf{nrl} \leftarrow S\,\overline{\mathsf{true}}])\,\mathsf{nrl} \\
= & \quad \langle \text{simplification of function update} \rangle \\
 & S\,\overline{\mathsf{true}} \Rightarrow S\,\overline{\mathsf{true}} \\
= & \quad \langle \text{lattice property} \rangle \\
 & \mathsf{true}
\end{array}$$

Thus with Theorem 8, $[\mathsf{tr}\ S]\,;S$ is universally conjunctive by definition. ■

Theorem 17 *Let S be an arbitrary positively conjunctive predicate transformer. Then there exists a unique predicate p and a unique indexed relation R such that $S = \{p\}\,;[R]$.*

Proof It is easily seen that we must choose $p = \mathsf{tr}\ S$:

$$\begin{array}{ll}
 & S = \{p\}\,;[R] \\
\Rightarrow & \quad \langle \text{congruence} \rangle \\
 & \mathsf{tr}\ S = \mathsf{tr}\,(\{p\}\,;[R]) \\
\equiv & \quad \langle \text{from definitions} \rangle \\
 & \mathsf{tr}\ S = (p \wedge \mathsf{tr}\,[R]) \\
\equiv & \quad \langle \text{from definitions} \rangle \\
 & \mathsf{tr}\ S = p
\end{array}$$

By Lemma 16 we have that $\{p\}\,;[R]$ is universally conjunctive. Then we can conclude by Theorem 15 that some indexed relation R exists such that $[\mathsf{tr}\,S]\,;S = [R]$, and by the proof of that theorem, $R\,x\,\sigma = (\forall P \mid S\,P\,\sigma \bullet P\,x)$. Then we have for arbitrary indexed predicate Q:

$$\begin{array}{ll} & (\{\mathsf{tr}\,S\}\,;S)\,Q \\ \equiv & \quad \langle\text{from definitions}\rangle \\ & (\{\mathsf{tr}\,S\}\,;[\mathsf{tr}\,S]\,;S)\,Q \\ \equiv & \quad \langle\text{definition of ;, assertion, assumption}\rangle \\ & \mathsf{tr}\,S \wedge (\mathsf{tr}\,S \Rightarrow S\,Q) \\ \equiv & \quad \langle\text{lattice property}\rangle \\ & \mathsf{tr}\,S \wedge S\,Q \\ \equiv & \quad \langle\text{definition of } \mathsf{tr}\,S\rangle \\ & S\,\overline{\mathsf{true}} \wedge S\,Q \\ \equiv & \quad \langle S \text{ is monotonic by Theorem 7}\rangle \\ & S\,Q \end{array}$$

Thus $S = \{p\}\,;[R]$ for p and R as above. ■

7 Conclusions

The results of this paper have been checked with the Isabelle/HOL theorem prover using a shallow embedding[1] [24]. The type of indexed state predicates, $\mathscr{P}\Sigma_{\mathscr{X}} = \mathscr{P}\Sigma_1 \times \cdots \times \mathscr{P}\Sigma_n$ can be expressed with a shallow embedding in higher order logic only for a specific value of n, but not for arbitrary n. We addressed this by two formalizations. One allows exits to have different state spaces, but is restricted to a small number of exits (up to three). This is used for the main results. A second formalization generalizes this to an arbitrary number of exits, but typing in higher order logic requires all Σ_i to be the same.

In conclusion, using the model of monotonic predicate transformers, we have explored the algebraic structure of multi-exit statements and we have given the normal forms of statements, positively conjunctive statements, and universally conjunctive statements. Like the normal form of single-exit statements, the normal form of Theorem 13 is non-constructive, in the sense that the exact weakest precondition $S\,P$ for each P is needed, which puts a limit in presence of recursion. Also, in the normal form $\{r\}\,;[R]$, the first statement "explodes" the state space and the second statement "shrinks" it again [25].

The normal form for positively conjunctive statements gives some insight for specification languages. Consider a procedure specification similar to those found in [5, 16]. We assume that the procedure operates on variables v:

$$\begin{array}{ll} \mathsf{requires} & p(v) \\ \mathsf{ensures} & a(v,v') \\ \mathsf{signals} & x : b(v,v') \\ \mathsf{signals} & y : c(v,v') \end{array}$$

Here, p is the precondition of the procedure and a,b,c are the "postconditions" for normal termination and when raising exceptions x,y. Such a specification can be understood as a statement S given by:

$$S \;=\; \{p\}\,;[\mathsf{nrl} \mapsto a, x \mapsto b, y \mapsto c]$$

[1]The Isabelle/HOL theory files are available at `http://www.cas.mcmaster.ca/~zhangt26/NormalForm`

This definition is in normal form for positively conjunctive statements. Hence this form allows an arbitrary positively conjunctive statement to be specified. We immediately have the $\mathsf{tr}\, S = p$, justifying calling p the precondition. We let $\mathsf{dom}\, r$ be the domain of relation r, formally $\mathsf{dom}\, r\, \sigma \equiv \exists \sigma' \bullet r\, \sigma\, \sigma'$. For the enabledness domain we calculate:

$$\begin{array}{ll} & \mathsf{en}\, S \\ = & \quad \langle \text{for any } p, S\text{: } \mathsf{en}\,(\{p\}\,;S) = (p \Rightarrow \mathsf{en}\ S) \rangle \\ & p \Rightarrow \mathsf{en}\,[\mathsf{nrl} \mapsto a, x \mapsto b, y \mapsto c] \\ = & \quad \langle \text{for any } R\text{: } \mathsf{en}\,[R] = \vee x \bullet \mathsf{dom}(R x) \rangle \\ & p \Rightarrow (\vee x \bullet \mathsf{dom}(R x)) \end{array}$$

If procedure calls are supposed to be always enabled, then this implies that under the precondition, at least one postcondition has to specify a final state. Hence this is a *feasibility condition* for specifications of multi-exit statements [19].

References

[1] Jean-Raymond Abrial (1996): *The B Book: Assigning Programs to Meanings*. Cambridge University Press.

[2] R. J. R. Back (1993): *Refinement Calculus, Lattices and Higher Order Logic*. In Manfred Broy, editor: Program Design Calculi, NATO ASI Series 118, Springer Berlin Heidelberg, pp. 53–71, doi:10.1007/978-3-662-02880-3_2.

[3] Ralph-Johan Back & Joakim von Wright (1998): *Refinement Calculus: A Systematic Introduction*. Springer-Verlag.

[4] R.J.R. Back & J. Wright (1990): *Refinement concepts formalised in higher order logic*. Formal Aspects of Computing 2(1), pp. 247–272, doi:10.1007/BF01888227.

[5] M. Barnett, K. R. M. Leino & W. Schulte (2005): *The Spec# Programming System: An Overview*. In G. Barthe, L. Burdy, M. Huisman, J.-L. Lanet & T. Muntean, editors: Construction and Analysis of Safe, Secure, and Interoperable Smart Devices, Springer, pp. 49–69. Available at `http://dx.doi.org/10.1007/978-3-540-30569-9_3`.

[6] Paulo Borba, Augusto Sampaio, Ana Cavalcanti & Márcio Cornélio (2004): *Algebraic reasoning for object-oriented programming*. Science of Computer Programming 52(1-3), pp. 53 – 100, doi:10.1016/j.scico.2004.03.003. Available at `http://www.sciencedirect.com/science/article/pii/S0167642304000474`.

[7] Flaviu Cristian (1984): *Correct and Robust Programs*. IEEE Transactions on Software Engineering 10(2), pp. 163–174.

[8] Michael J.C. Gordon (1989): *Mechanizing Programming Logics in Higher Order Logic*. In Graham Birtwistle & P.A. Subrahmanyam, editors: Current Trends in Hardware Verification and Automated Theorem Proving, Springer New York, pp. 387–439, doi:10.1007/978-1-4612-3658-0_10.

[9] Matthew Hennessy & Robin Milner (1985): *Algebraic laws for nondeterminism and concurrency*. J. ACM 32(1), pp. 137–161. Available at `http://doi.acm.org.libaccess.lib.mcmaster.ca/10.1145/2455.2460`.

[10] C. A. R. Hoare, I. J. Hayes, He Jifeng, C. C. Morgan, A. W. Roscoe, J. W. Sanders, I. H. Sorensen, J. M. Spivey & B. A. Sufrin (1987): *Laws of Programming*. Communications of the ACM 30(8), pp. 672–686. Available at `http://doi.acm.org/10.1145/27651.27653`.

[11] C. A. R. Hoare, He Jifeng & A. Sampaio (1993): *Normal form approach to compiler design*. Acta Informatica 30(8), pp. 701–739. Available at `http://dx.doi.org/10.1007/BF01191809`.

[12] Bart Jacobs (2001): *A Formalisation of Java's Exception Mechanism.* In D. Sands, editor: *ESOP '01: Proceedings of the 10th European Symposium on Programming Languages and Systems*, Springer-Verlag, pp. 284–301.

[13] He Jifeng, Ian Page & Jonathan Bowen (1993): *Towards a Provably Correct Hardware Implementation of Occam.* In George Milne & Laurence Pierre, editors: *Correct Hardware Design and Verification Methods*, *Lecture Notes in Computer Science* 683, Springer Berlin / Heidelberg, pp. 214–225. Available at `http://dx.doi.org/10.1007/BFb0021726`.

[14] Steve King & Carroll Morgan (1995): *Exits in the Refinement Calculus.* *Formal Aspects of Computing* 7(1), pp. 54–76.

[15] Dexter Kozen (1997): *Kleene Algebra with Tests.* *ACM Trans. Program. Lang. Syst.* 19(3), pp. 427–443. Available at `http://doi.acm.org.libaccess.lib.mcmaster.ca/10.1145/256167.256195`.

[16] G. T. Leavens, A. L. Baker & C. Ruby (2006): *Preliminary design of JML: a behavioral interface specification language for Java.* *SIGSOFT Software Engineering Notes* 31, pp. 1–38. Available at `http://doi.acm.org/10.1145/1127878.1127884`.

[17] Jing Li, Huibiao Zhu & Jifeng He (2007): *Algebraic Semantics for Compensable Transactions.* In Cliff Jones, Zhiming Liu & Jim Woodcock, editors: *Theoretical Aspects of Computing – ICTAC 2007*, *Lecture Notes in Computer Science* 4711, Springer Berlin / Heidelberg, pp. 306–321. Available at `http://dx.doi.org/10.1007/978-3-540-75292-9_21`.

[18] Bertrand Meyer (1997): *Object-Oriented Software Construction*, 2nd edition. Prentice-Hall.

[19] Carroll C. Morgan (1994): *Programming from Specifications*, 2nd edition. Prentice Hall.

[20] A.W. Roscoe & C.A.R. Hoare (1988): *The laws of OCCAM programming.* *Theoretical Computer Science* 60(2), pp. 177 – 229, doi:10.1016/0304-3975(88)90049-7. Available at `http://www.sciencedirect.com/science/article/pii/0304397588900497`.

[21] Emil Sekerinski & Tian Zhang (2012): *Verification Rules for Exception Handling in Eiffel.* In Rohit Gheyi & David Naumann, editors: *Formal Methods: Foundations and Applications*, *Lecture Notes in Computer Science* 7498, Springer Berlin / Heidelberg, pp. 179–193.

[22] Emil Sekerinski & Tian Zhang (2013): *On a New Notion of Partial Refinement.* In John Derrick, Eerke Boiten & Steve Reeves, editors: *Proceedings 16th International Refinement Workshop*, *Electronic Proceedings in Theoretical Computer Science* 115, Open Publishing Association, pp. 1–14, doi:10.4204/EPTCS.115.1.

[23] Leila Silva, Augusto Sampaio & Edna Barros (2004): *A Constructive Approach to Hardware/Software Partitioning.* *Formal Methods in System Design* 24(1), pp. 45–90. Available at `http://dx.doi.org/10.1023/B:FORM.0000004787.52329.98`.

[24] Markus Wenzel (2011): *The Isabelle/Isar Reference Manual.* Available at `http://isabelle.in.tum.de/doc/isar-ref.pdf`.

[25] J. von Wright (1994): *The lattice of data refinement.* *Acta Informatica* 31(2), pp. 105–135, doi:10.1007/BF01192157.

[26] M. Ying (2003): *Reasoning about probabilistic sequential programs in a probabilistic logic.* *Acta Informatica* 39(5), pp. 315–389. Available at `http://dx.doi.org/10.1007/s00236-003-0113-z`.