A Single Complete Refinement Rule for Demonic Specifications

Karl Lermer[1] and Paul Strooper[2]

[1] Software Verification Research Centre
[2] Dept. of Comp. Science and Elec. Eng.,
The University of Queensland, Queensland 4072, Australia,
{lermer,pstroop}@csee.uq.edu.au

Abstract. We present the complete lattice of demonic languages and its interpretation in refinement proofs. In contrast to the conventional approach of refinement with an abstraction relation on the underlying state spaces, we introduce a notion of refinement with an abstraction relation on the power sets of the state spaces. This allows us to derive a single complete refinement rule for demonic specifications.

1 Introduction

In [10], a refinement semantics is presented for so-called demonic specifications that lifts the refinement of operations defined in formal specifiation languages such as Z and VDM to the refinement of state machines. The semantics is consistent with the conventional refinement semantics for the system underlying a specification in Z [14]. Rather than using a system or state machine semantics via the subset ordering of prefix-closed languages [5,6,14], an approach is taken where operations are interpreted as relations on state spaces, and input/output histories are used to define the semantics of the underlying state machine. This approach is not state dependent, as is the case for the improved failure model of CSP [4,6]. Instead, the languages and specifications are restricted to so-called demonic ones, where the enabling on a new input is dependent only on the past input history independent of the past outputs. The refinement relation is not simply trace inclusion where traces may disappear during the refinement process; it requires the refined system to accept the same or more traces as the original one. As usual, refinement relations on the operational level are defined via abstraction relations between the underlying state spaces. In analogy to standard simulation techniques [5,9,6,11,14,13], it is then possible to express every refinement with the help of two refinement methods on the operational level, denoted by *forward* and *backward* refinement.

A relational approach with abstraction relations as above will always necessitate two refinement rules for completeness. By using a predicate-transformer semantics, Gardiner and Morgan [2] obtain completeness of refinement by using a single refinement technique, called *cosimulation* (a predicate transformer with certain properties).

In this paper, we build on the relational approach of [10] to further investigate the completeness of refinement. As refinement semantics we will use the partial ordering on *prefix-closed* and *demonic* languages introduced in [10]. We then extend the results from [10], by proving that this ordering defines a natural lattice structure on the prefix-closed and demonic languages, and discuss its importance for refinement proofs. This is in analogy to the subset ordering on languages and its interpretation in refinements [3,1,5,11,14,13]. Furthermore, we generalise the notion of refinement relations via abstraction relations on state spaces to refinement relations via abstraction relations on power sets of state spaces. As for the predicate transformer setting [2], it is then possible to derive a single complete refinement rule.

In Section 2 we introduce the notions of *prefix-closed* and *demonic* languages, and *demonic* specifications from [10]. In Section 3, we present the lattice structure on these languages, and in Section 4 we introduce a notion of refinement on specifications and show its soundness and completeness. We conclude in Section 5 with a comparison with related work.

2 Languages and Specifications

We view a module as a black box, accessible only through a fixed set of *operations* — the exported procedures and functions. A *module interface specification* (hereafter just *specification*) specifies the behaviour of the module. The *syntax* of the specification states the names of the access routines, and their inputs and outputs. We use Op to denote the set of all operation names, In to denote the set of all inputs, and Out to denote the set of all outputs.

The *semantics* of the specification describes the observable behaviour of the operations. We are interested in comparing the behaviour of different specifications. Because there are many ways to represent the state in a specification, we need a definition of behaviour that is independent of the state representation. We first consider *histories* (observable behaviours): finite, possibly empty sequences of the form

$$h = \langle c_1, v_1\rangle\langle c_2, v_2\rangle \dots \langle c_n, v_n\rangle$$

For $i \in \{1, ..., n\}$, $c_i = \langle \iota_i, op_i\rangle$ is a call to an operation $op_i \in Op$ with input $\iota_i \in In$, and $v_i \in Out$ is an output. If an operation has no input or output, we use the special symbol $\perp$ to indicate this. We use the symbol ε to denote the empty history.

2.1 Languages

The set of all histories, $\mathcal{H}$, is determined by Op, In, and Out. A language $\mathcal{L}$ is defined as a subset of $\mathcal{H}$. We only consider non-empty languages that are *prefix-closed*: for any history $h \in \mathcal{L}$ and any call-value pair $\langle c, v\rangle$, if $h\langle c, v\rangle \in \mathcal{L}$ then $h \in \mathcal{L}$.

$$\mathcal{L}an_{\mathcal{H}} = \{\mathcal{L} \subseteq \mathcal{H} \mid \mathcal{L} \neq \varnothing \wedge \mathcal{L} \text{ is prefix-closed}\}$$

This implies $\varepsilon \in \mathcal{L}$ for all languages in $\mathcal{L}an_{\mathcal{H}}$. We introduce the following operators on histories. For any history

$$h = \langle c_1, v_1\rangle\langle c_2, v_2\rangle \ldots \langle c_n, v_n\rangle$$

we denote the corresponding *trace* or *input sequence* by

$$\mathcal{I}(h) = c_1 c_2 \ldots c_n$$

As for histories, we use ε to denote the empty trace and thus $\mathcal{I}(\varepsilon) = \varepsilon$. For a set of histories $H \subseteq \mathcal{H}$, we define the set of all traces of H by

$$Tr(H) = \{\mathcal{I}(h) \mid h \in H\}$$

For a language $\mathcal{L}$ and a trace $t \in Tr(\mathcal{H})$, we collect all possible histories with trace t in the set

$$\Omega_{\mathcal{L}}(t) = \{h \mid h \in \mathcal{L} \wedge \mathcal{I}(h) = t\}$$

We will also use the following operators on finite sequences $\sigma = s_1 s_2 ... s_n$: $front(\sigma) = s_1 s_2 ... s_{n-1}$, $last(\sigma) = s_n$, and $\sigma \uparrow m = s_1 ... s_m$. Finally, we use $\#\sigma$ to denote the length of σ.

In [10], we show that the class of *demonic* languages provides a natural semantics for data refinement in VDM and Z. Intuitively, a language $\mathcal{L}$ is demonic if it is prefix-closed and the fact that an input is enabled depends solely on the past input history, independent of the corresponding outputs.

Definition 1. *A language $\mathcal{L}$ is demonic if*

a) $\mathcal{L}$ is prefix-closed.
b) $\forall\, h_1, h_2 \in \mathcal{L} \,:\, \mathcal{I}(h_1) = \mathcal{I}(h_2) \Rightarrow$
$\quad \forall\, \iota \in In, \omega \in Out, op \in Op \,:$
$\quad h_1\langle\langle \iota, op\rangle, \omega\rangle \in \mathcal{L} \Rightarrow \exists\, \omega' \in Out \,:\, h_2\langle\langle \iota, op\rangle, \omega'\rangle \in \mathcal{L}$

The set of demonic languages will be denoted by

$$\mathcal{L}an^d_{\mathcal{H}} = \{\mathcal{L} \subseteq \mathcal{H} \mid \mathcal{L} \neq \varnothing \wedge \mathcal{L} \text{ is demonic}\}$$

For instance, deterministic languages ($\forall\, \tau \in Tr(\mathcal{L}) \,:\, \#\Omega_{\mathcal{L}}(\tau) = 1$) and total languages ($\forall\, h \in \mathcal{L}, \iota \in In, op \in Op\, \exists\, \omega \in Out \,:\, h\langle\langle \iota, op\rangle, \omega\rangle \in \mathcal{L}$) are demonic.

A useful characterisation for demonic languages is the following: a language $\mathcal{L}$ is demonic iff

$$\forall\, \tau \in Tr(\mathcal{L}) \setminus \{\varepsilon\} \,:\, \Omega_{\mathcal{L}}(front(\tau)) = \{\, h \uparrow (\#\tau - 1) \mid h \in \Omega_{\mathcal{L}}(\tau)\,\}$$

Unfortunately, the set of demonic languages $\mathcal{L}an^d_{\mathcal{H}}$ does not behave as nicely as the set of prefix-closed languages $\mathcal{L}an_{\mathcal{H}}$, which forms a complete lattice under the inclusion ordering $\subseteq$ and the usual set operations. In general, demonic languages are not closed under intersection and union. We will see below that $\mathcal{L}an^d_{\mathcal{H}}$ carries a lattice structure under a different ordering relation.

We introduce a partial ordering $\Subset$ on languages. In Section 3, we use the poset (partially ordered set)

$$(\mathcal{L}an^{d}_{\mathcal{H}}, \Subset)$$

to define a lattice structure on $\mathcal{L}an^{d}_{\mathcal{H}}$, and in Section 4, we use this lattice as a domain for the characterisation of refinement proofs.

Definition 2. *Let $\mathcal{L}$ and $\mathcal{L}'$ be languages in $\mathcal{H}$,*

$$\mathcal{L}' \Subset \mathcal{L} \ \ \textit{iff} \ \ Tr(\mathcal{L}) \subseteq Tr(\mathcal{L}') \ \wedge \ \forall\, t \in Tr(\mathcal{L}) \ :\ \Omega_{\mathcal{L}'}(t) \subseteq \Omega_{\mathcal{L}}(t)$$

For languages $\mathcal{L}$ and $\mathcal{L}'$, $\mathcal{L}' \Subset \mathcal{L}$ if all traces in $\mathcal{L}$ also occur in $\mathcal{L}'$, and if every history in $\mathcal{L}'$ corresponding to a trace in $\mathcal{L}$ is also a history in $\mathcal{L}$.

2.2 Specifications

A specification S defines a language $\mathcal{L}$ — the subset of $\mathcal{H}$ expressing the behaviour defined by the specification. In general the form of the specification may vary, but in this paper we focus on *model-based* specifications, where the behaviour is specified in terms of a state space St.

Definition 3. *A (model-based) specification S is a six-tuple*

$$(Op, St, In, Out, \textsc{Init}, \_^{S})$$

with operation (name) set Op, state set St, input set In, output set Out, a nonempty set of initial states $\textsc{Init} \subseteq St$, and an interpretation function

$$\_^{S} \ :\ Op \to \mathbb{P}\,((In \times St) \times (St \times Out))$$

Note that Op, St, In, Out, $\textsc{Init}$ can be infinite sets. Any operation $op \in Op$ is interpreted via $\_^{S}$ as a set of pairs

$$(\langle \iota, s\rangle, \langle s', \omega\rangle)$$

where each pair represents a state transition with input $\iota \in In$, internal states $s, s' \in St$ (s denotes the state before and s' the state after the operation is performed), and output $\omega \in Out$.

For a specification S, the *precondition* of operation $op \in Op$ with input $\iota \in In$ will be denoted by

$$pre_S(\langle \iota, op\rangle) = \{s \in St \mid \exists\, s' \in St, \omega \in Out \ :\ (\langle \iota, s\rangle, \langle s', \omega\rangle) \in op^{S}\,\}$$

We recursively define the *postcondition* of a trace $t \in Tr(\mathcal{H})$ by

$$ptrace_S(t) = \begin{cases} \textsc{Init} & \text{if } t \text{ is the empty trace} \\ \{s' \in St \mid \exists\, s \in St, \omega \in Out \ : & \\ \qquad (\langle \iota, s\rangle, \langle s', \omega\rangle) \in op^{S} \wedge s \in ptrace_S(t_1)\} & \\ \qquad \text{if } t = t_1\langle \iota, op\rangle & \end{cases}$$

Note that in our setting, pre- and postconditions denote sets of states, not predicates. Given a specification S and a history $h \in \mathcal{H}$, we denote the set of *final states* of h by

$$\mathit{final}_S(h) = \begin{cases} \textsc{Init} & \text{if } h = \varepsilon \\ \{s' \in \mathit{St} \mid \exists\, s \in \mathit{St} \,:\, (\langle \iota, s\rangle, \langle s', \omega\rangle) \in \mathit{op}^S \wedge s \in \mathit{final}_S(h_1)\} \\ & \text{if } h = h_1 \langle\langle \iota, \mathit{op}\rangle, \omega\rangle \end{cases}$$

We can now define the language accepted by a specification S, consisting of the empty history and all histories that are produced by starting from an initial state in $\textsc{Init}$ and recursively applying the operations from Op.

Definition 4. *For a specification S, the language accepted by S is*

$$\begin{aligned} \mathcal{L}_S = \{h \in \mathcal{H} \mid\, & h = \varepsilon \vee \exists\, h_1 \in \mathcal{L}_S, \mathit{op} \in \mathit{Op}, \iota \in \mathit{In}, \omega \in \mathit{Out} \,: \\ & h = h_1\langle\langle \iota, \mathit{op}\rangle, \omega\rangle \wedge (\exists\, s \in \mathit{final}_S(h_1), s' \in \mathit{St} \,:\, (\langle \iota, s\rangle, \langle s', \omega\rangle) \in \mathit{op}^S)\} \end{aligned}$$

It follows from this definition that $\mathcal{L}_S$ is prefix-closed (i.e., $\mathcal{L}_S \in \mathcal{L}\mathit{an}_{\mathcal{H}}$).

There is a notion corresponding to demonic languages for specifications. A specification S is demonic if whenever an input/operation pair $\langle \iota, \mathit{op}\rangle$ is enabled in a final state of a history $h \in \mathcal{L}_S$ it must be enabled in all the final states of the trace $\mathcal{I}(h)$. Again, the input enabling depends only on the input history.

Definition 5. *A specification S is demonic if*

$$\forall\, \tau \in \mathit{Tr}(\mathcal{L}_S) \setminus \{\varepsilon\} \;:\; \mathit{ptraces}_S(\mathit{front}(\tau)) \subseteq \mathit{pre}_S(\mathit{last}(\tau))$$

Every demonic specification S defines a demonic language $\mathcal{L}_S$ [10, Prop. 2]. The converse is not true in general: there are non-demonic specifications that specify demonic languages. However, for every demonic language $\mathcal{L}$ there exists a demonic specification that defines $\mathcal{L}$ [10, Prop. 11].

As an example of a demonic specification, consider the following random number generating specification $SA1$ from [10].

There are two operations: *random* generates a random integer value and has no output, and *val* returns the value generated by the last call to *random* as its output. If no call to *random* has been made, *val* returns 0.

$$\begin{aligned} \mathit{Op} &= \{\mathit{random}, \mathit{val}\}\,,\; \mathit{Out} \;=\; \mathbb{Z} \cup \{\perp\}\,,\; \mathit{In} \;=\; \{\perp\} \\ \mathit{St}^{SA1} &= \mathbb{Z}\,,\; \textsc{Init}^{SA1} = \{0\} \\ \mathit{random}^{SA1} &= \{(\langle \perp, s\rangle, \langle s', \perp\rangle) \mid s, s' \in \mathbb{Z}\} \\ \mathit{val}^{SA1} &= \{(\langle \perp, i\rangle, \langle i, i\rangle) \mid i \in \mathbb{Z}\} \end{aligned}$$

By adding the operation $\mathit{two} \;=\; \{(\langle \perp, 2\rangle, \langle 2, 2\rangle)\}$ to specification $SA1$ we obtain a non-demonic specification. This is because *two* is only enabled at state 2 and not on all states that can be generated by *random*.

To motivate the use of demonic specifications and languages and to give a few generic examples, we state the correspondence to the commonly used failure and trace models in the theory of communicating sequential processes (CSP) [6, 7].

Given a specification $S = (\mathit{Op}, \mathit{St}, \mathit{In}, \mathit{Out}, \textsc{Init}, \_^S)$ and a fresh symbol ζ we define two derived demonic specifications, the total completion

$$S(TC) = (\mathit{Op}, \mathit{St} \cup \{\zeta\}\ \mathit{In}, \mathit{Out} \cup \{\zeta\}, \textsc{Init}, \_^{S(TC)})$$

and the failure completion

$$S(FC) = (\mathit{Op}, \mathit{St} \cup \{\zeta\}, \mathit{In}, \mathit{Out} \cup \mathbb{P}(\mathit{In} \times \mathit{Op} \times \mathit{Out}), \textsc{Init}, \_^{S(FC)})$$

of S. For every operation symbol $op \in \mathit{Op}$ we define

$$\begin{aligned} op^{S(TC)} &= op^S \cup \{(\langle \iota, s\rangle, \langle \zeta, \zeta\rangle) \,:\, \iota \in \mathit{In} \wedge s \in \mathit{St} \cup \{\zeta\}\} \\ op^{S(FC)} &= op^S \cup \{(\langle \iota, s\rangle, \langle \zeta, X\rangle) \,:\, \iota \in \mathit{In} \wedge s \in \mathit{St} \wedge X \in \mathbb{P}(\mathit{In} \times \mathit{Op} \times \mathit{Out}) \\ &\qquad\qquad \wedge \exists\, \omega \in \mathit{Out} \,:\, \langle\langle \iota, op\rangle, \omega\rangle \in X \\ &\qquad\qquad \wedge \forall \langle\langle \iota_0, op_0\rangle, \omega_0\rangle \in X \,\nexists s' \in \mathit{St} \,:\, (\langle \iota_0, s\rangle, \langle s', \omega_0\rangle) \in {op_0}^S \,\} \\ &\qquad \cup \{(\langle \iota, \zeta\rangle, \langle \zeta, \varnothing\rangle) \,:\, \iota \in \mathit{In}\} \end{aligned}$$

In both specifications the state ζ is interpreted as the state that the system enters after a failure occurred. The total completion extends the behaviour of the original specification by assuming that a failure can occur in any state, no matter what the input is: the system may enter the failure state at any moment. The failure completion handles failures in a more sophisticated manner. A failure transition can only occur if the transition was impossible in the original system. With the help of the set X the modified system can output failure transitions at any state.

The total and the failure completion of a specification S are demonic specifications.

Proposition 1. *For every specification S, $S(TC)$ and $S(FC)$ are demonic with total languages $\mathcal{L}_{S(TC)}$ and $\mathcal{L}_{S(FC)}$.*

Defining the failures of a specification S by

$$\begin{aligned} \mathit{failures}(S) = \{(h, X) \,:\, h \in \mathcal{L}_S \wedge X \in \mathbb{P}(\mathit{In} \times \mathit{Op} \times \mathit{Out}) \\ \wedge \exists\, s \in \mathit{final}_S(h)\, \forall \langle\langle \iota, op\rangle, \omega\rangle \in X \nexists s' \in \mathit{St} \,:\, (\langle \iota, s\rangle, \langle s', \omega\rangle) \in op^S\} \end{aligned}$$

we can formulate the following theorem. It shows that prominent failure and trace models of CSP [6,7] can be expressed via demonic specifications and the ordering relation $\Subset$. For brevity, and because the theorem is not used later on, the proofs have been omitted from the paper. The first equivalence is rather obvious and states that the inclusion relation on the languages (input/output trace sets) of the specifications is characterised by the $\Subset$ relation of the underlying total completions. The second equivalence characterises the inclusion relation on the failure sets of specifications in terms of the $\Subset$ relation on the underlying failure completions.

Theorem 1. *Let S^A and S^C be any specifications. Then, the following correspondences hold.*

i) $\mathcal{L}_{S^C(TC)} \Subset \mathcal{L}_{S^A(TC)} \Leftrightarrow \mathcal{L}_{S^C} \subseteq \mathcal{L}_{S^A}$
ii) $\mathcal{L}_{S^C(FC)} \Subset \mathcal{L}_{S^A(FC)} \Leftrightarrow \mathit{failures}(S^C) \subseteq \mathit{failures}(S^A)$

3 Lattice of Demonic Languages

In this section, we present the lattice structure that is induced on the prefix-closed and demonic languages by the ordering relation $\Subset$. First we introduce operators that define the supremum and infimum in the lattice. To obtain a complete lattice, we add the new symbol $\perp$ as the smallest element to both $\mathcal{L}an_{\mathcal{H}}$ and $\mathcal{L}an^d_{\mathcal{H}}$.

Definition 6. *i)* $\mathcal{L}_\varepsilon = \{\varepsilon\}$, ${\mathcal{L}an_{\mathcal{H}}}^* = \mathcal{L}an_{\mathcal{H}} \cup \{\perp\}$, ${\mathcal{L}an^d_{\mathcal{H}}}^* = \mathcal{L}an^d_{\mathcal{H}} \cup \{\perp\}$
ii) For a nonempty family of languages $\mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}$, $i \in I$*:*

$$IT(\mathcal{L}_i) = \cap_{i \in I} Tr(\mathcal{L}_i)\,,\ UT(\mathcal{L}_i) = \cup_{i\in I} Tr(\mathcal{L}_i)$$

iii) For a nonempty family of languages $\mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}$, $i \in I$*:*

$$\begin{aligned}
\vee_I\, \mathcal{L}_i &= \cup\{\Omega_{\mathcal{L}_i}(\tau) \mid i \in I \wedge \tau \in IT(\mathcal{L}_i)\}\\
\wedge_I\, \mathcal{L}_i &= \begin{cases} \perp \quad \textit{if } \exists \tau \in UT(\mathcal{L}_i) : \cap\{\Omega_{\mathcal{L}_i}(\tau) \mid i \in I \wedge \tau \in Tr(\mathcal{L}_i)\} = \varnothing \\ \cup\{\cap\{\Omega_{\mathcal{L}_i}(\tau) \mid i \in I \wedge \tau \in Tr(\mathcal{L}_i)\} \mid \tau \in UT(\mathcal{L}_i)\} \quad \textit{otherwise}\end{cases}\\
\wedge^p_I\, \mathcal{L}_i &= \begin{cases} \perp \quad \textit{if } \nexists \mathcal{L} \in \mathcal{L}an_{\mathcal{H}} : \mathcal{L} \subseteq \wedge_I\, \mathcal{L}_i \wedge Tr(\mathcal{L}) = Tr(\wedge_I\, \mathcal{L}_i) \\ \cup\{\mathcal{L} \in \mathcal{L}an_{\mathcal{H}} \mid \mathcal{L} \subseteq \wedge_I\, \mathcal{L}_i \wedge Tr(\mathcal{L}) = Tr(\wedge_I\, \mathcal{L}_i)\} \quad \textit{otherwise}\end{cases}\\
\wedge^d_I\, \mathcal{L}_i &= \begin{cases} \perp \quad \textit{if } \nexists \mathcal{L} \in \mathcal{L}an^d_{\mathcal{H}} : \mathcal{L} \subseteq \wedge_I\, \mathcal{L}_i \wedge Tr(\mathcal{L}) = Tr(\wedge_I\, \mathcal{L}_i) \\ \cup\{\mathcal{L} \in \mathcal{L}an^d_{\mathcal{H}} \mid \mathcal{L} \subseteq \wedge_I\, \mathcal{L}_i \wedge Tr(\mathcal{L}) = Tr(\wedge_I\, \mathcal{L}_i)\} \quad \textit{otherwise}\end{cases}
\end{aligned}$$

It is possible that for prefix-closed languages $\mathcal{L}_i$, $i \in I$, the language $\wedge_I\, \mathcal{L}_i$ is not prefix-closed. In fact, we will see that $\wedge^p_I\, \mathcal{L}_i$, if not $\perp$, is the greatest prefix-closed language below $\wedge_I\, \mathcal{L}_i$ (w.r.t $\Subset$). Also, if all $\mathcal{L}_i$, $i \in I$ are demonic then $\wedge_I\, \mathcal{L}_i$ is not necessarily demonic. This is illustrated by the following example. Assume operations c_1 and c_2 with no inputs (we will use c_1 as a shorthand for the call $\langle \perp, c_1\rangle$, and similarly for c_2), and outputs o_1, o_2, o_3, and o_4. We define the languages

$$\begin{aligned}
\mathcal{L}_1 = \{&\varepsilon, \langle\langle c_1, o_1\rangle\rangle, \langle\langle c_1, o_2\rangle\rangle, \langle\langle c_2, o_3\rangle\rangle, \langle\langle c_2, o_4\rangle\rangle,\\
&\langle\langle c_1, o_1\rangle\langle c_2, o_3\rangle\rangle, \langle\langle c_1, o_2\rangle\langle c_2, o_3\rangle\rangle, \langle\langle c_2, o_3\rangle\langle c_1, o_2\rangle\rangle, \langle\langle c_2, o_4\rangle\langle c_1, o_2\rangle\rangle\}\\
\mathcal{L}_2 = \{&\varepsilon, \langle\langle c_1, o_1\rangle\rangle, \langle\langle c_2, o_3\rangle\rangle, \langle\langle c_2, o_4\rangle\rangle, \langle\langle c_2, o_3\rangle\langle c_1, o_1\rangle\rangle, \langle\langle c_2, o_4\rangle\langle c_1, o_2\rangle\rangle\}
\end{aligned}$$

$\mathcal{L}_1$, $\mathcal{L}_2$ are demonic languages with

$$\begin{aligned}
\mathcal{L}_1 \wedge \mathcal{L}_2 = \{&\varepsilon, \langle\langle c_1, o_1\rangle\rangle, \langle\langle c_2, o_3\rangle\rangle, \langle\langle c_2, o_4\rangle\rangle,\\
&\langle\langle c_1, o_1\rangle\langle c_2, o_3\rangle\rangle, \boxed{\langle\langle c_1, o_2\rangle\langle c_2, o_3\rangle\rangle}, \langle\langle c_2, o_4\rangle\langle c_1, o_2\rangle\rangle\}\\
\mathcal{L}_1 \wedge^p \mathcal{L}_2 = \{&\varepsilon, \langle\langle c_1, o_1\rangle\rangle, \boxed{\langle\langle c_2, o_3\rangle\rangle}, \langle\langle c_2, o_4\rangle\rangle,
\end{aligned}$$

$$\langle\langle c_1, o_1\rangle\langle c_2, o_3\rangle\rangle, \langle\langle c_2, o_4\rangle\langle c_1, o_2\rangle\rangle\}$$
$$\mathcal{L}_1 \wedge^d \mathcal{L}_2 = \{\varepsilon, \langle\langle c_1, o_1\rangle\rangle, \langle\langle c_2, o_4\rangle\rangle, \langle\langle c_1, o_1\rangle\langle c_2, o_3\rangle\rangle, \langle\langle c_2, o_4\rangle\langle c_1, o_2\rangle\rangle\}$$

The following two lemmas prove properties of $\wedge_I \mathcal{L}_i$, $\wedge_I^p \mathcal{L}_i$, $\wedge_I^d \mathcal{L}_i$, and $\vee_I \mathcal{L}_i$ that will allow us to derive the complete lattice structure for languages ordered by $\Subset$.

Lemma 1 *For any nonempty family of languages $\mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}$, $i \in I$:*
a) $\wedge_I^p \mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}{}^*$ and $\wedge_I^d \mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}^{d\,*}$
b) If $\wedge_I \mathcal{L}_i \neq \perp$ then $\wedge_I \mathcal{L}_i \Subset \mathcal{L}_j$ for all $j \in I$

Proof. a) The union of prefix-closed languages is again prefix-closed, and the union of demonic languages with the same set of traces is also demonic [10, Prop. 1].

b) For any $j \in I$, $Tr(\mathcal{L}_j) \subseteq UT(\mathcal{L}_i) = Tr(\wedge_I \mathcal{L}_i)$ and for $\tau \in Tr(\mathcal{L}_j)$ we get $\Omega_{\wedge_I \mathcal{L}_i}(\tau) = \cap\{\Omega_{\mathcal{L}_i}(\tau) \mid i \in I \wedge \tau \in Tr(\mathcal{L}_i)\} \subseteq \Omega_{\mathcal{L}_j}(\tau)$ □

Lemma 2 *For any nonempty family of languages $\mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}$, $i \in I$:*
a) $\vee_I \mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}$
b) If all $\mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}^d$, then $\vee_I \mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}^d$

Proof. a) Let $hz \in \vee_I \mathcal{L}_i$. Then, $\mathcal{I}(hz) \in IT(\mathcal{L}_i)$ and since $\mathcal{L}_i$, $i \in I$, is prefix-closed, we can conclude $\mathcal{I}(h) \in IT(\mathcal{L}_i)$. Moreover, for all $i_0 \in I$ for which $hz \in \mathcal{L}_{i_0}$, because $\mathcal{L}_{i_0}$ is prefix-closed, we have $h \in \mathcal{L}_{i_0}$ and so, by definition of the $\vee$-operator, $h \in \vee_I \mathcal{L}_i$.

b) Let $\mathcal{L}_i$, $i \in I$ be demonic. Then we fix a trace $\tau \in Tr(\vee_I \mathcal{L}_i) \setminus \{\varepsilon\}$ and some $h \in \Omega_{\vee_I \mathcal{L}_i}(front(\tau))$. Because

$$\Omega_{\vee_I \mathcal{L}_i}(front(\tau)) = \cup\{\Omega_{\mathcal{L}_i}(front(\tau)) \mid i \in I\}$$

we can find $i_0 \in I$ with $h \in \Omega_{\mathcal{L}_{i_0}}(front(\tau))$. Since $\mathcal{L}_{i_0}$ is demonic and $\tau \in \cap_{i\in I} Tr(\mathcal{L}_i)$ there exists a $h' \in \Omega_{\mathcal{L}_{i_0}}(\tau)$ with $h' \uparrow (\#\tau - 1) = h$. We can conclude $h' \in \Omega_{\vee_I \mathcal{L}_i}(\tau)$ and so $\vee_I \mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}^d$. □
We also need the following result, proven in [10, Prop. 5].

Lemma 3 *For languages $\mathcal{L}, \mathcal{L}' \subseteq \mathcal{H}$ we have:*
a) $\mathcal{L}' \Subset \mathcal{L} \Rightarrow Tr(\mathcal{L} \cap \mathcal{L}') = Tr(\mathcal{L}) \cap Tr(\mathcal{L}') = Tr(\mathcal{L})$, $\mathcal{L}' \Subset \mathcal{L}' \cap \mathcal{L}$
b) $(\mathcal{L}'$ demonic $\wedge\ \mathcal{L}' \Subset \mathcal{L}) \Rightarrow \mathcal{L}' \cap \mathcal{L}$ demonic

If we extend the ordering $\Subset$ in a canonical way from $\mathcal{L}an_{\mathcal{H}}$ to $\mathcal{L}an_{\mathcal{H}}{}^*$ such that $\perp$ becomes the smallest element in $\mathcal{L}an_{\mathcal{H}}{}^*$, then we do obtain a complete lattice. Furthermore, we set $\vee_\varnothing \mathcal{L}_i = \perp$ and $\wedge_\varnothing \mathcal{L}_i = \wedge_\varnothing^p \mathcal{L}_i = \wedge_\varnothing^d \mathcal{L}_i = \mathcal{L}_\varepsilon$.

Theorem 2. *a) $(\mathcal{L}an_{\mathcal{H}}{}^*, \Subset)$ is a complete lattice with greatest element $\mathcal{L}_\varepsilon$ and smallest element $\perp$. For a family $\mathcal{L}_i$, $i \in I$ of languages in $\mathcal{L}an_{\mathcal{H}}$, its supremum is $\vee_{\{i\in I \mid \mathcal{L}_i \neq \perp\}} \mathcal{L}_i$ and its infimum is $\perp$ if there is at least one $i \in I$ with $\mathcal{L}_i = \perp$ and $\wedge_I^p \mathcal{L}_i$ otherwise.*

b) $({\mathcal{L}an^{d}_{\mathcal{H}}}^{*}, \Subset)$ *is a complete lattice with greatest element* $\mathcal{L}_\varepsilon$ *and smallest element* $\bot$*. For a family* $\mathcal{L}_i$*,* $i \in I$ *of languages in* $\mathcal{L}an^{d}_{\mathcal{H}}$*, its supremum is* $\vee_{\{i\in I \mid \mathcal{L}_i \neq \bot\}} \mathcal{L}_i$ *and its infimum is* $\bot$ *if there is at least one* $i \in I$ *with* $\mathcal{L}_i = \bot$ *and* $\wedge^{d}_{I} \mathcal{L}_i$ *otherwise.*

Proof. a) Let $\mathcal{L} \in \mathcal{L}an_{\mathcal{H}}$. Then, $\{\varepsilon\} = Tr(\mathcal{L}_\varepsilon) \subseteq Tr(\mathcal{L})$ and $\Omega_{\mathcal{L}}(\varepsilon) = \{\varepsilon\} \subseteq \Omega_{\mathcal{L}_\varepsilon}(\varepsilon)$. Hence, $\mathcal{L} \Subset \mathcal{L}_\varepsilon$ which shows that $\mathcal{L}_\varepsilon$ is the greatest element in ${\mathcal{L}an_{\mathcal{H}}}^{*}$.

Let $\mathcal{L}_i$, $i \in I$ be a nonempty family of languages in $\mathcal{L}an_{\mathcal{H}}$. Applying Lemma 2 a) shows $\vee_I \mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}$. For any $j \in I$, $Tr(\vee_I \mathcal{L}_i) = IT(\mathcal{L}_i) \subseteq Tr(\mathcal{L}_j)$. For any trace $\tau \in Tr(\vee_I \mathcal{L}_i)$,

$$\Omega_{\mathcal{L}_j}(\tau) \subseteq \cup_{i\in I} \Omega_{\mathcal{L}_i}(\tau) = \Omega_{\vee_I \mathcal{L}_i}(\tau)$$

Therefore, $\mathcal{L}_j \Subset \vee_I \mathcal{L}_i$, $j \in I$.

Moreover, for any $\mathcal{L} \in \mathcal{L}an_{\mathcal{H}}$ with $\mathcal{L}_i \Subset \mathcal{L}$, we have $Tr(\mathcal{L}) \subseteq IT(\mathcal{L}_i) = Tr(\vee_I \mathcal{L}_i)$ and for $\tau \in Tr(\mathcal{L})$ we have

$$\Omega_{\vee_I \mathcal{L}_i}(\tau) = \cup_{i\in I} \Omega_{\mathcal{L}_i}(\tau) \subseteq \cup_{i\in I} \Omega_{\mathcal{L}}(\tau) = \Omega_{\mathcal{L}}(\tau)$$

We deduce $\vee_I \mathcal{L}_i \Subset \mathcal{L}$.

This proves that $\vee_I \mathcal{L}_i$ is the least upper bound of the family $\mathcal{L}_i$, $i \in I$ in $\mathcal{L}an_{\mathcal{H}}$. It remains to extend this in a straightforward manner to ${\mathcal{L}an_{\mathcal{H}}}^{*}$.

Next we claim $\wedge_I \mathcal{L}_i \Subset \mathcal{L}_j$ for every $j \in I$. This is trivial if $\wedge_I \mathcal{L}_i = \bot$ and the remaining case has been shown in Lemma 1 b).

Therefore, $\wedge^{p}_{I} \mathcal{L}_i \subseteq \wedge_I \mathcal{L}_i \Subset \mathcal{L}_j$ for every $j \in I$. Since $\wedge^{p}_{I} \mathcal{L}_i$, if not equal to $\bot$, has the same set of traces as $\wedge_I \mathcal{L}_i$, we can infer

$$\wedge^{p}_{I} \mathcal{L}_i \Subset \wedge_I \mathcal{L}_i \Subset \mathcal{L}_j$$

for every $j \in I$. Let $\mathcal{L} \in {\mathcal{L}an_{\mathcal{H}}}^{*}$ with $\mathcal{L} \Subset \mathcal{L}_j$, for all $j \in I$. We must prove that $\mathcal{L} \Subset \wedge^{p}_{I} \mathcal{L}_i$, which trivially holds for $\mathcal{L} = \bot$. For $\mathcal{L} \neq \bot$, we first prove that $\mathcal{L} \Subset \wedge_I \mathcal{L}_i$. Note that $\varnothing \neq Tr(\wedge_I \mathcal{L}_i) = UT(\mathcal{L}_i) \subseteq Tr(\mathcal{L})$, Moreover, $Tr(\mathcal{L}_j) \subseteq Tr(\mathcal{L})$ for all $j \in I$ and, if $\tau \in UT(\mathcal{L}_i)$ then

$$\Omega_{\mathcal{L}}(\tau) \subseteq \cap\{\Omega_{\mathcal{L}_i}(\tau) \mid i \in I \wedge \tau \in Tr(\mathcal{L}_i)\} = \Omega_{\wedge_I \mathcal{L}_i}(\tau)$$

Hence, $\mathcal{L} \Subset \wedge_I \mathcal{L}_i$.

We define $\mathcal{L}' = \mathcal{L} \cap (\wedge_I \mathcal{L}_i)$ and claim the following two properties.

$$\mathcal{L}' \in \mathcal{L}an_{\mathcal{H}} \tag{1}$$

$$\mathcal{L} \Subset \mathcal{L}' \Subset \wedge^{p}_{I} \mathcal{L}_i \Subset \wedge_I \mathcal{L}_i \tag{2}$$

We show (1) first. Note that $\varepsilon \in \mathcal{L}'$. Let $hz \in \mathcal{L}'$. $\mathcal{L}$ is prefix-closed and so $h \in \mathcal{L}$. Furthermore, $\mathcal{I}(hz) \in Tr(\wedge_I \mathcal{L}_i) = UT(\mathcal{L}_i)$. From $\mathcal{L}_i$, $i \in I$ being prefix-closed, we can deduce $\mathcal{I}(h) \in Tr(\wedge_I \mathcal{L}_i)$. Since $\mathcal{L} \Subset \wedge_I \mathcal{L}_i$, we can conclude $h \in \wedge_I \mathcal{L}_i$ and therefore $h \in \mathcal{L}'$.

To prove (2), we note that $\wedge^{p}_{I} \mathcal{L}_i \Subset \wedge_I \mathcal{L}_i$, which was shown above. $\mathcal{L}'$ is prefix-closed by (1), and furthermore $\mathcal{L}' \subseteq \wedge_I \mathcal{L}_i$ and $\mathcal{L} \Subset \wedge_I \mathcal{L}_i$. By Lemma 3 a), we can conclude $Tr(\mathcal{L}') = Tr(\wedge_I \mathcal{L}_i)$. Hence, by definition of the $\wedge^p$-operator we obtain $\mathcal{L}' \subseteq \wedge_I^p \mathcal{L}_i$. That $\mathcal{L}'$ and $\wedge_I^p \mathcal{L}_i$ have the same traces then implies

$$\mathcal{L}' \Subset \wedge_I^p \mathcal{L}_i$$

The property $\mathcal{L} \Subset \mathcal{L}'$ follows from Lemma 3 a) and $\mathcal{L} \Subset \wedge_I \mathcal{L}_i$.

From (2) we obtain $\mathcal{L} \Subset \wedge_I^p \mathcal{L}_i$ by transitivity of $\Subset$, which is what we wanted to verify.

b) It is obvious that $\mathcal{L}_\varepsilon$ is demonic and since $\mathcal{L}_\varepsilon$ is the greatest element in $\mathcal{L}an_{\mathcal{H}}{}^*$, it follows that $\mathcal{L}_\varepsilon$ is also the greatest element in $\mathcal{L}an_{\mathcal{H}}^{d}{}^*$.

For any family $\mathcal{L}_i$, $i \in I$ in $\mathcal{L}an_{\mathcal{H}}^d$ we get $\vee_I \mathcal{L}_i$ as its supremum in $\mathcal{L}an_{\mathcal{H}}$. According to Lemma 2 b), $\vee_I \mathcal{L}_i$ is demonic and therefore, it is the supremum of $\mathcal{L}_i$, $i \in I$ in $\mathcal{L}an_{\mathcal{H}}^d$.

It remains to show that $\wedge_I^d \mathcal{L}_i$ is the greatest lower bound for $\mathcal{L}_i$, $i \in I$ in $\mathcal{L}an_{\mathcal{H}}^d$. From the definitions of $\wedge_I^p \mathcal{L}_i$ and $\wedge_I^d \mathcal{L}_i$ we can infer $\wedge_I^d \mathcal{L}_i \Subset \wedge_I^p \mathcal{L}_i$ and together with a) we then obtain

$$\wedge_I^d \mathcal{L}_i \Subset \wedge_I^p \mathcal{L}_i \Subset \mathcal{L}_j \, , \, j \in I$$

Now, assume $\mathcal{L} \in \mathcal{L}an_{\mathcal{H}}^d$ with $\mathcal{L} \Subset \mathcal{L}_i$, for all $i \in I$. With a) we deduce

$$\mathcal{L} \Subset \wedge_I^p \mathcal{L}_i \Subset \wedge_I \mathcal{L}_i$$

As above, by setting $\mathcal{L}' = \mathcal{L} \cap (\wedge_I \mathcal{L}_i)$ and by using Lemma 3 b), we obtain that $\mathcal{L}'$ is demonic. Property (2) holds again and by definition of $\wedge_I^d \mathcal{L}_i$,

$$\mathcal{L}' \subseteq \wedge_I^d \mathcal{L}_i$$

Hence, $\mathcal{L} \Subset \mathcal{L}' \subseteq \wedge_I^d \mathcal{L}_i$. Note that $Tr(\wedge_I^d \mathcal{L}_i) = Tr(\wedge_I \mathcal{L}_i) = Tr(\mathcal{L}')$ which follows from Lemma 3 a). Therefore,

$$\mathcal{L} \Subset \mathcal{L}' \Subset \wedge_I^d \mathcal{L}_i$$

and from the transitivity of $\Subset$ we can conclude that $\mathcal{L} \Subset \wedge_I^d \mathcal{L}_i$. □

The following results further explore the correspondence between the operators $\wedge$, $\wedge^p$ and $\wedge^d$.

Proposition 1 *If $\mathcal{L}_i \in \mathcal{L}an_{\mathcal{H}}$, $i \in I$, is a downward ordered net (i.e., $\forall\, i,j \in I \; \exists\, k \in I \; : \; \mathcal{L}_k \Subset \mathcal{L}_i \wedge \mathcal{L}_k \Subset \mathcal{L}_j$), then $\wedge_I \mathcal{L}_i = \wedge_I^p \mathcal{L}_i$.*

Proof. For a downward-ordered net $\mathcal{L}_i$, $i \in I$, since $\wedge_I^p \mathcal{L}_i$, if not equal to $\perp$, is the greatest prefix-closed set in $\wedge_I \mathcal{L}_i$, we simply have to show that $\wedge_I \mathcal{L}_i$ is prefix-closed.

Let $hz \in \wedge_I \mathcal{L}_i$. There is $i_0 \in I$ such that $hz \in \mathcal{L}_{i_0}$. $\mathcal{L}_{i_0}$ is prefix-closed, and so $h \in \mathcal{L}_{i_0}$. Thus, $\mathcal{I}(h) \in Tr(\wedge_I \mathcal{L}_i)$.

Now for all $i_1 \in I$ such that $\mathcal{I}(h) \in Tr(\mathcal{L}_{i_1})$, there exists an $i_2 \in I$ with $\mathcal{L}_{i_2} \Subset \mathcal{L}_{i_0}$ and $\mathcal{L}_{i_2} \Subset \mathcal{L}_{i_1}$. Hence, $\mathcal{I}(hz) \in Tr(\mathcal{L}_{i_2})$ and $hz \in \mathcal{L}_{i_2}$. $\mathcal{L}_{i_2}$ is prefix-closed and so $h \in \mathcal{L}_{i_2}$. From $\Omega_{\mathcal{L}_{i_2}}(\mathcal{I}(h)) \subseteq \Omega_{\mathcal{L}_{i_1}}(\mathcal{I}(h))$ we conclude $h \in \mathcal{L}_{i_1}$.

From this we conclude $h \in \wedge_I \mathcal{L}_i$, which proves that $\wedge_I \mathcal{L}_i$ is prefix-closed. □

Definition 7. *A language $\mathcal{L}$ is τ-output-finite for a trace $\tau \in Tr(\mathcal{L}) \setminus \{\varepsilon\}$ if for all histories $h \in \Omega_{\mathcal{L}}(front(\tau))$ the set*

$$\{h' \mid h' \in \Omega_{\mathcal{L}}(\tau) \wedge h' \uparrow \#h = h\}$$

is finite. $\mathcal{L}$ is output-finite if it is τ-output-finite for all $\tau \in Tr(\mathcal{L}) \setminus \{\varepsilon\}$.

Proposition 2 *If $\mathcal{L}_i \in \mathcal{L}an^d_{\mathcal{H}}$, $i \in I$ is a downward-ordered net and for all $\tau \in Tr(\wedge_I \mathcal{L}_i)$ there exists an $i \in I$ with $\mathcal{L}_i$ being τ-output-finite, then $\wedge_I \mathcal{L}_i = \wedge^d_I \mathcal{L}_i$.*

Proof. It suffices to prove that $\wedge_I \mathcal{L}_i$ is demonic, assuming $\wedge_I \mathcal{L}_i \neq \perp$. Let $\tau \in Tr(\wedge_I \mathcal{L}_i) \setminus \{\varepsilon\}$ and $h \in \Omega_{\wedge_I \mathcal{L}_i}(front(\tau))$. We find $i_1 \in I$ with $\mathcal{L}_{i_1}$ being τ-output finite. Furthermore, $\mathcal{L}_{i_1}$ is demonic, and so we can conclude $front(\tau) \in Tr(\mathcal{L}_{i_1})$ and that the set

$$\mathcal{A} = \{h' \mid h' \in \Omega_{\mathcal{L}_{i_1}}(\tau) \wedge h' \uparrow \#h = h\}$$

is nonempty and finite. We claim that there exists an $h' \in \mathcal{A}$ with $h' \in \wedge_I \mathcal{L}_i$. If not, we can find $j_1, ..., j_n \in I$ such that $\tau \in \cap^n_{k=1} Tr(\mathcal{L}_{j_k})$ and $\cap^n_{k=1} \Omega_{\mathcal{L}_{j_k}}(\tau) \cap \mathcal{A} = \varnothing$. But there exists an $i_2 \in I$ with $\mathcal{L}_{i_2} \Subset \mathcal{L}_{i_1}$ and $\mathcal{L}_{i_2} \Subset \mathcal{L}_{j_k}$, $1 \leq k \leq n$. We infer $\tau \in Tr(\mathcal{L}_{i_2})$ and $h \in \mathcal{L}_{i_2}$. Again, $\mathcal{L}_{i_2}$ is demonic and so

$$\varnothing \neq \{h' \mid h' \in \Omega_{\mathcal{L}_{i_2}}(\tau) \wedge h' \uparrow \#h = h\} \subseteq \cap^n_{k=1} \Omega_{\mathcal{L}_{j_k}}(\tau) \cap \mathcal{A}$$

which contradicts the assumption above. □

This shows that $\wedge_I \mathcal{L}_i$ is demonic, and hence $\wedge_I \mathcal{L}_i = \wedge^d_I \mathcal{L}_i$, for downward-ordered nets of output-finite and demonic languages $\mathcal{L}_i$, $i \in I$.

4 Refinement

Data-refinement proofs [3,1,8,14,13] are used to verify that a specification (or implementation) S^C with a concrete state representation is correct with respect to a specification S^A with an abstract state representation. In [10], we show the soundness and completeness of refinement with respect to the ordering $\Subset$ on demonic languages using a combination of forward and backward refinement. Here we generalise this notion of refinement to one that is based on an abstraction relation on the power sets of the underlying state spaces, and show that this provides a single sound and complete refinement rule.

Definition 8. *Given two specifications $S^A = (Op, St^A, In, Out, \textsc{Init}^A, \_^{S^A})$ and $S^C = (Op, St^C, In, Out, \textsc{Init}^C, \_^{S^C})$, an abstraction relation*

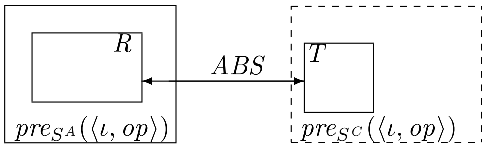
$$ABS : \mathbb{P}\, St^C \leftrightarrow \mathbb{P}\, St^A$$

and operation $op \in Op$, we say that op^{S^A} power-refines to op^{S^C} ($op^{S^A} \sqsubseteq_{ABS} op^{S^C}$) if the following conditions hold.

$(P1)\ \forall \iota \in \mathit{In},\ R \in \mathbb{P}\,\mathit{St}^A,\ T \in \mathbb{P}\,\mathit{St}^C :$
$\quad ((T, R) \in \mathit{ABS} \wedge \varnothing \neq R \subseteq \mathit{pre}_{S^A}(\langle \iota, \mathit{op}\rangle)) \Rightarrow\ T \subseteq \mathit{pre}_{S^C}(\langle \iota, \mathit{op}\rangle))$

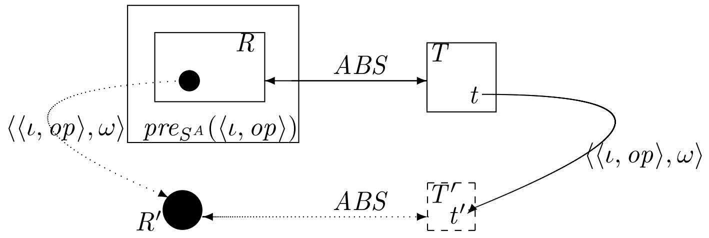
$(P2)\ \forall \iota \in \mathit{In},\ \omega \in \mathit{Out},\ R \in \mathbb{P}\,\mathit{St}^A,\ T \in \mathbb{P}\,\mathit{St}^C,\ t \in T, t' \in \mathit{St}^C :$
$\quad ((T, R) \in \mathit{ABS} \wedge \varnothing \neq R \subseteq \mathit{pre}_{S^A}(\langle \iota, \mathit{op}\rangle) \wedge (\langle \iota, t\rangle, \langle t', \omega\rangle) \in \mathit{op}^{S^C}) \Rightarrow$
$\quad (\exists\, R' \in \mathbb{P}\,\mathit{St}^A,\ T' \in \mathbb{P}\,\mathit{St}^C :$
$\quad (R' \neq \varnothing \wedge (T', R') \in \mathit{ABS} \wedge t' \in T'$
$\quad \wedge\ (\forall\, s_1 \in R'\, \exists\, s_0 \in R : (\langle \iota, s_0\rangle, \langle s_1, \omega\rangle) \in \mathit{op}^{S^A}))$

Refinement with properties $(P1)$ and $(P2)$ lifts the common forward refinement of VDM and Z [8,10,14] with abstraction relations on the state spaces to refinement with abstraction relations on power sets. $(P1)$ states that if T and R are related via ABS and $\langle \iota, \mathit{op}\rangle$ is enabled in all states in R then $\langle \iota, \mathit{op}\rangle$ is enabled in all states in T (denoted by the dashed lines in the picture below).



R
ABS
T
$\mathit{pre}_{S^A}(\langle \iota, \mathit{op}\rangle)$
$\mathit{pre}_{S^C}(\langle \iota, \mathit{op}\rangle)$

$(P2)$ states that via ABS every input of op^{S^A} must be accepted by op^{S^C} with outputs that were possible for op^{S^A}. The last condition in $(P2)$ asserts that all states that are contained in the abstract state set R' and related to a concrete state set T' are final states under the abstract operation with input and output that were accepted by the concrete operation. This is illustrated in the picture below where the dashed lines indicate how the concrete transition is simulated by the abstract one.



R
ABS
T
t
$\langle\langle \iota, \mathit{op}\rangle, \omega\rangle$
$\mathit{pre}_{S^A}(\langle \iota, \mathit{op}\rangle)$
$\langle\langle \iota, \mathit{op}\rangle, \omega\rangle$
R'
ABS
T'
t'

This notion of operation refinement naturally leads to the following notion of specification refinement.

Definition 9. *We say that specification* $S^A = (\mathit{Op}, \mathit{St}^A, \mathit{In}, \mathit{Out}, \textsc{Init}^A, \_^{S^A})$ *power-refines to specification* $S^C = (\mathit{Op}, \mathit{St}^C, \mathit{In}, \mathit{Out}, \textsc{Init}^C, \_^{S^C})$ *and write* $S^A \sqsubseteq S^C$ *if there exists an abstraction relation* ABS *as above such that*

(PR1) $\forall\, \mathit{op} \in \mathit{Op} : \mathit{op}^{S^A} \sqsubseteq_{\mathit{ABS}} \mathit{op}^{S^C}$
(PR2) $(\textsc{Init}^C, \textsc{Init}^A) \in \mathit{ABS}$

We write $S^A \sqsubseteq_{ABS} S^C$ if we want to explicitly indicate the abstraction relation ABS.

We will see below that the relation $\sqsubseteq$ defines a preorder on demonic specifications. Obligation ($PR1$) states that every abstract operation can be power-refined to a concrete one and obligation ($PR2$) states that the abstract initial state set corresponds to the concrete initital state set. Note that we overload the semantics of the symbol $\sqsubseteq$. It will be obvious from the context whether we mean operation or specification refinement.

As an example, consider the following alternative specification $SA2$ of the random number generating specification $SA1$ of Section 2.2.

$$\begin{aligned} St^{SA2} &= \mathbb{Z} \cup \{\mathit{undef}\}\,,\ \textsc{Init}^{SA2} = \{0\} \\ \mathit{random}^{SA2} &= \{(\langle \bot, s_0\rangle, \langle \mathit{undef}, \bot\rangle) \mid s_0 \in \mathbb{Z} \cup \{\mathit{undef}\}\} \\ \mathit{val}^{SA2} &= \{(\langle \bot, i\rangle, \langle i, i\rangle) \mid i \in \mathbb{Z}\} \cup \{(\langle \bot, \mathit{undef}\rangle, \langle i, i\rangle) \mid i \in \mathbb{Z}\} \end{aligned}$$

In specification $SA1$ the operation *random* generates a random integer number and a subsequent call to *val* makes this number visible. In $SA2$, *random* will always set the state to *undef* and only a subsequent call to *val* will generate a random number. In [10], we show that there is a backward refinement from $SA1$ to $SA2$, but no forward refinement. However, there is a power-refinement such that $SA1$ power-refines to $SA2$. An abstraction relation that shows this is

$$ABS = \{(\{i\}, \{i\}) \ : \ i \in \mathbb{Z}\} \cup \{(\{\mathit{undef}\}, \mathbb{Z})\}$$

If we use the ordering $\Subset$ as the underlying semantics of specification refinement and power-refinement with obligations ($PR1$) and ($PR2$) as the refinement technique, then Theorem 3 below states the soundness and completeness of power-refinement for demonic specifications.

Theorem 3. *For demonic specifications S^A and S^C, $\mathcal{L}_{S^C} \Subset \mathcal{L}_{S^A} \Leftrightarrow S^A \sqsubseteq S^C$.*

Proof. To prove $\mathcal{L}_{S^C} \Subset \mathcal{L}_{S^A} \Rightarrow S^A \sqsubseteq S^C$, we use the abstraction relation

$$(T, R) \in ABS \text{ iff } \exists\, h \in \mathcal{L}_{S^C} \cap \mathcal{L}_{S^A} \ : \ T = \mathit{final}_{S^C}(h) \wedge R = \mathit{final}_{S^A}(h)$$

Because $\varepsilon \in \mathcal{L}_{S^C} \cap \mathcal{L}_{S^A}$, $\textsc{Init}^C = \mathit{final}_{S^C}(\varepsilon)$, and $\textsc{Init}^A = \mathit{final}_{S^A}(\varepsilon)$, we can conclude $(\textsc{Init}^C, \textsc{Init}^A) \in ABS$, which establishes ($PR2$).

For ($P1$), we assume $(T, R) \in ABS$ and $\varnothing \neq R \subseteq \mathit{pre}_{S^A}(\langle \iota, op\rangle)$. According to the definition of ABS, there is $h \in \mathcal{L}_{S^C} \cap \mathcal{L}_{S^A}$ with $T = \mathit{final}_{S^C}(h)$ and $R = \mathit{final}_{S^A}(h)$. From $\mathcal{I}(h)\langle \iota, op\rangle \in Tr(\mathcal{L}_{S^A})$ and $\mathcal{L}_{S^C} \Subset \mathcal{L}_{S^A}$, it follows $\mathcal{I}(h)\langle \iota, op\rangle \in Tr(\mathcal{L}_{S^C})$. S^C is demonic and therefore $T = \mathit{final}_{S^C}(h) \subseteq \mathit{ptrace}_{S^C}(\mathcal{I}(h)) \subseteq \mathit{pre}_{S^C}(\langle \iota, op\rangle)$.

To prove ($P2$), we additionally assume $\omega \in \mathit{Out}$, $t \in T$ and $t' \in St^C$ with $(\langle \iota, t\rangle, \langle t', \omega\rangle) \in op^{S^C}$. Hence, $h\langle\langle \iota, op\rangle, \omega\rangle \in \mathcal{L}_{S^C}$ and because of $\mathcal{L}_{S^C} \Subset \mathcal{L}_{S^A}$ we have $h\langle\langle \iota, op\rangle, \omega\rangle \in \mathcal{L}_{S^A}$. For $T' = \mathit{final}_{S^C}(h\langle\langle \iota, op\rangle, \omega\rangle)$ and $R' = \mathit{final}_{S^A}(h\langle\langle \iota, op\rangle, \omega\rangle)$ we derive $(T', R') \in ABS$, $t' \in T'$ and $R' \neq \varnothing$. Finally, because $R' = \mathit{final}_{S^A}(h\langle\langle\iota, op\rangle, \omega\rangle)$, for any $s_1 \in R'$ there is a $s_0 \in R$ such that $(\langle\iota, s_0\rangle, \langle s_1, \omega\rangle) \in op^{S^A}$, which concludes the proof of $(P2)$.

To prove $\mathcal{L}_{S^C} \Subset \mathcal{L}_{S^A} \Leftarrow S^A \sqsubseteq S^C$, we first prove the following property by induction on the length of the traces in $Tr(\mathcal{L}_{S^A})$.

$$\begin{array}{l}
\forall\, \tau = \langle\iota_1, op_1\rangle ... \langle\iota_{\#\tau}, op_{\#\tau}\rangle \in Tr(\mathcal{L}_{S^A}) \\
\;\exists\;\, R_0, ..., R_{\#\tau},\, T_0, ..., T_{\#\tau}, t_0, ..., t_{\#\tau}, \omega_1, ..., \omega_{\#\tau} \;: \\
\;a)\; R_0 = \textsc{Init}^A \wedge T_0 = \textsc{Init}^C \\
\;b)\; \forall\; 0 \le i \le \#\tau \;:\; t_i \in T_i \wedge (T_i, R_i) \in ABS \\
\;c)\; \forall\; 1 \le i \le \#\tau \;:\; (\langle\iota_i, t_{i-1}\rangle, \langle t_i, \omega_i\rangle) \in {op_i}^{S^C} \\
\;d)\; \forall\, 0 \le i \le \#\tau \;:\; \varnothing \ne R_i \subseteq \mathit{ptrace}_{S^A}(\tau \uparrow i) \\
\;e)\; \forall\, 1 \le i \le \#\tau\, \forall\, s_1 \in R_i\, \exists\, s_0 \in R_{i-1} \;:\; (\langle\iota_i, s_0\rangle, \langle s_1, \omega_i\rangle) \in {op_i}^{S^A}
\end{array} \qquad (3)$$

Base case ($\tau = \varepsilon$): We have $(\textsc{Init}^C, \textsc{Init}^A) \in ABS$ and $\mathit{ptrace}_{S^A}(\varepsilon) = \textsc{Init}^A \ne \varnothing$ by definition. Similarly, we can select $t_0 \in T_0 = \textsc{Init}^C \ne \varnothing$.

Induction step ($\tau' = \tau\langle\iota, op\rangle \in Tr(\mathcal{L}_{S^A})$): For $\tau \in Tr(\mathcal{L}_{S^A})$ we can find $R_0, ..., R_{\#\tau},\, T_0, ..., T_{\#\tau}, t_0, ..., t_{\#\tau}, \omega_1, ..., \omega_{\#\tau}$ that satisfy property (3) according to our induction hypothesis. Since S^A is demonic, it follows from property (3) d) that $\varnothing \ne R_{\#\tau} \subseteq \mathit{ptrace}_{S^A}(\tau) \subseteq \mathit{pre}_{S^A}(\langle\iota, op\rangle)$. From $(P1)$ and $(T_{\#\tau}, R_{\#\tau}) \in ABS$ we can infer $T_{\#\tau} \subseteq \mathit{pre}_{S^C}(\langle\iota, op\rangle)$. Then we find $t' \in St^C$, $\omega \in Out$ such that $(\langle\iota, t_{\#\tau}\rangle, \langle t', \omega\rangle) \in op^{S^C}$. Property $(P2)$ then supplies us with $R' \in \mathbb{P}\, St^A$, $T' \in \mathbb{P}\, St^C$ such that $R' \ne \varnothing$, $(T', R') \in ABS$ and $t' \in T'$. Furthermore, for any $s_1 \in R'$ there is a $s_0 \in R_{\#\tau}$ with $(\langle\iota, s_0\rangle, \langle s_1, \omega\rangle) \in op^{S^A}$. We know by our induction hypothesis that $R_{\#\tau} \subseteq \mathit{ptrace}_{S^A}(\tau)$ and therefore $R' \subseteq \mathit{ptrace}_{S^A}(\tau')$. The induction step is completed by setting $R_{\#\tau'} = R'$, $T_{\#\tau'} = T'$, $t_{\#\tau'} = t'$ and $\omega_{\#\tau'} = \omega$.

We proceed by proving the following property by induction on the length of the traces in $Tr(\mathcal{L}_{S^C}) \cap Tr(\mathcal{L}_{S^A})$.

$$\begin{array}{l}
\forall\, h = \langle\langle\iota_1, op_1\rangle, \omega_1\rangle ... \langle\langle\iota_{\#h}, op_{\#h}\rangle, \omega_{\#h}\rangle \in \mathcal{L}_{S^C} \;:\; \mathcal{I}(h) \in Tr(\mathcal{L}_{S^A}) \Rightarrow \\
\forall\, t_0, ..., t_{\#h} \in St^C \;: \\
(t_0 \in \textsc{Init}^C \wedge (\forall\, 1 \le i \le \#h \;:\; (\langle\iota_i, t_{i-1}\rangle, \langle t_i, \omega_i\rangle) \in {op_i}^{S^C})) \Rightarrow \\
\exists\, R_0, ..., R_{\#h},\, T_0, ..., T_{\#h} \;: \\
\;a)\; R_0 = \textsc{Init}^A \wedge T_0 = \textsc{Init}^C \\
\;b)\; \forall\; 0 \le i \le \#h \;:\; t_i \in T_i \wedge (T_i, R_i) \in ABS \\
\;c)\; \forall\, 0 \le i \le \#h \;:\; \varnothing \ne R_i \subseteq \mathit{final}_{S^A}(h \uparrow i) \\
\;d)\; \forall\, 1 \le i \le \#h\, \forall\, s_1 \in R_i\, \exists\, s_0 \in R_{i-1} \;:\; (\langle\iota_i, s_0\rangle, \langle s_1, \omega_i\rangle) \in {op_i}^{S^A}
\end{array} \qquad (4)$$

Base case ($h = \varepsilon$): This trivially holds because $\mathit{final}_{S^A}(\varepsilon) = \textsc{Init}^A \ne \varnothing$ and $(\textsc{Init}^C, \textsc{Init}^A) \in ABS$.

Induction step ($h' = h\langle\langle \iota_{\#h'}, op_{\#h'}\rangle, \omega_{\#h'}\rangle \in \mathcal{L}_{S^C}$ with $\mathcal{I}(h') \in Tr(\mathcal{L}_{S^A})$): Let $t_0 \in \textsc{Init}^C$, $t_1, ..., t_{\#h'} \in St^C$ with $(\langle \iota_i, t_{i-1}\rangle, \langle t_i, \omega_i\rangle) \in {op_i}^{S^C}$, $1 \leq i \leq \#h'$. By the induction hypothesis there are sets $R_0, ..., R_{\#h}$, $T_0, ..., T_{\#h}$ that satisfy the property (4). That is $\varnothing \neq R_{\#h} \subseteq \mathit{final}_{S^A}(h) \subseteq \mathit{ptrace}_{S^A}(\mathcal{I}(h))$, $(T_{\#h}, R_{\#h}) \in ABS$ and $\mathcal{I}(h)\langle \iota_{\#h'}, op_{\#h'}\rangle \in Tr(\mathcal{L}_{S^A})$. The specification S^A is demonic and therefore $R_{\#h} \subseteq \mathit{ptrace}_{S^A}(\mathcal{I}(h)) \subseteq \mathit{pre}_{S^A}(\langle \iota_{\#h'}, op_{\#h'}\rangle)$. With $(P2)$ we then can find $R' \in \mathbb{P}\, St^A$, $T' \in \mathbb{P}\, St^C$ such that $R' \neq \varnothing$, $(T', R') \in ABS$ and $t_{\#h'} \in T'$. Furthermore, for any $s_1 \in R'$ there is a $s_0 \in R_{\#h}$ with $(\langle \iota_{\#h'}, s_0\rangle, \langle s_1, \omega_{\#h'}\rangle) \in {op_{\#h'}}^{S^A}$. Since $R_{\#h} \subseteq \mathit{final}_{S^A}(h)$ we can conclude that $R' \subseteq \mathit{final}_{S^A}(h')$. The induction step is completed by setting $R_{\#h'} = R'$ and $T_{\#h'} = T'$.

Finally, note that property (3) implies $Tr(\mathcal{L}_{S^A}) \subseteq Tr(\mathcal{L}_{S^C})$ and from property (4) we can infer $\Omega_{\mathcal{L}_{S^C}}(\tau) \subseteq \Omega_{\mathcal{L}_{S^A}}(\tau)$ for every trace $\tau \in Tr(\mathcal{L}_{S^A})$. This implies $\mathcal{L}_{S^C} \Subset \mathcal{L}_{S^A}$. $\square$

As a corollary, power-refinement $\sqsubseteq$ defines a preorder on the set of demonic specifications.

5 Conclusions and Related Work

In this paper, we have extended the results on the refinement of demonic languages from [10]. We have presented a lattice structure for the set of demonic languages and proved the correspondence to the set of demonic specifications. By lifting the common notion of abstraction relations on state spaces to relations on the power sets of state spaces, it was possible to derive a single complete refinement rule. This refinement rule provides a characterisation of the ordering on demonic languages in terms of a refinement preorder on demonic specifications. This complements the classical characterisation by backward and forward refinement proven in [10].

The refinement ordering we use is consistent with the conventional semantics underlying a system specification in Z and VDM [14,13]. Hence the forward and backward refinement techniques of these specification languages are sound and complete methods for the verification of the refinement ordering $\Subset$ on demonic languages. To our knowledge, there is no refinement theory founded on abstraction relations on power sets of states. All relational approaches we are aware of work with abstraction relations on state spaces and therefore need at least two refinement rules for completeness results [5,9,7,6,11,14,13] or significant restrictions on the languages and specifications. Furthermore, those approaches are founded on the subset ordering on prefix-closed languages.

The restriction to demonic specifications and the generalisation to abstraction relations on power sets make it possible to obtain a similar completeness result as in the predicate transformer setting [2] with only one refinement technique. As for predicate transformers, finding abstraction relations with a single complete rule is in general more difficult than relying on the combination of two less complex rules. Hence for practical applications the combination of backward and forward refinement rules is recommended. Comparing the predicate transformer approach with our approach, Rewitzky and Brink [12] show that predicate transformers can be interpreted as total functions on the power sets of the state spaces. The predicate transformer refinement ordering can then be seen as the subset ordering on the underlying behaviours, which implies that traces can disappear during the refinement process. Any cosimulation may be seen as a total function on the power sets of the state spaces. Our refinement ordering on demonic languages and specifications differs from the subset ordering in that traces cannot disappear during the refinement. Furthermore, a demonic composition principle in specifications is more general than the composition by total operations. Also, in our relational approach, a restriction to abstraction functions on power sets would be insufficient.

Acknowledgements: We thank Ian Hayes and the anonymous referees for their suggestions on earlier versions of the paper.

References

1. R. J. R. Back. On correct refinement of programs. *Journal of Computer and System Sciences*, 23:49–68, 1981.
2. P. H. B. Gardiner and C. Morgan. A single complete rule for data refinement. *Formal Aspects of Computing*, 5(4):367–382, 1993.
3. C.A.R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1(4):271–281, 1972.
4. C.A.R. Hoare. *Communicating sequential processes.* Prentice-Hall International, UK, LTD, 1985.
5. C.A.R. Hoare, He Jifeng, and J. W. Sanders. Prespecifications in data refinement. *Information Processing Letters*, 25:71–76, 1987.
6. He Jifeng. Process simulation and refinement. *Formal Aspects of Computing*, 1:229–241, 1989.
7. He Jifeng. Various simulations and refinements. In J.W. de Bakker, C., W.P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, volume 430 of *Lecture Notes in Computer Science*, pages 340–360. Springer-Verlag, 1989.
8. C.B. Jones. *Systematic Software Development Using VDM.* Prentice-Hall, second edition, 1990.
9. M. B. Josephs. A state-based approach to communicating processes. *Distributed Computing*, 3:9–18, 1988.
10. K. Lermer and P. Strooper. Refinement and state machine abstraction. Technical Report 00-01, Software Verification Research Centre, January 2000. To appear in *Theoretical Computer Science.*
11. N. Lynch and F. Vaandrager. Forward and backward simulation for timing-based systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 397–445. Springer-Verlag, 1991.
12. I. Rewitzky and C. Brink. Predicate transformers as power operations. *Formal Aspects of Computing*, 7(2):169–182, 1995.
13. W.-P. Roever and K. Engelhardt. *Data refinement: model-oriented proof methods and their comparison.* Cambridge tracts in theoretical computer science; 4. Cambridge University Press, 1998.
14. J. Woodcock and J. Davies. *Using Z: Specification, Refinement, and Proof.* Prentice-Hall, 1996.