Acta Cybernetica **15** (2002) 379–416.

An Arithmetic Theory of Consistency Enforcement

Sebastian Link* and Klaus-Dieter Schewe*

Abstract

Consistency enforcement starts from a given program specification S and a static invariant $\mathcal{I}$ and aims to replace S by a slightly modified program specification $S_{\mathcal{I}}$ that is provably consistent with respect to $\mathcal{I}$. One formalization which suggests itself is to define $S_{\mathcal{I}}$ as the greatest consistent specialization of S with respect to $\mathcal{I}$, where specialization is a partial order on semantic equivalence classes of program specifications.

In this paper we present such a theory on the basis of arithmetic logic. We show that with mild technical restrictions and mild restrictions concerning recursive program specifications it is possible to obtain the greatest consistent specialization gradually and independently from the order of given invariants as well as by replacing basic commands by their respective greatest consistent specialization. Furthermore, this approach allows to discuss computability and decidability aspects for the first time.

1 Introduction

In order to capture the semantics of a system, almost all approaches to formal specification provide at least static invariants. Then the problem is to guarantee consistency. For a program specification S and an invariant $\mathcal{I}$ this means that every execution of S starting in a state that satisfies $\mathcal{I}$ should always lead to a state satisfying $\mathcal{I}$, too. This is usually relaxed so that only terminating executions of S are considered, in which case the problems of termination and of consistency can be handled separately.

If program semantics is expressed axiomatically by the use of predicate transformers leading to weakest (liberal) preconditions, then consistency leads to the well known proof obligation $\mathcal{I} \Rightarrow wlp(S)(\mathcal{I})$. Verification of such proof obligations can then be a very hard task.

As an alternative consistency enforcement has been considered. In particular, in the field of databases, where the complexity of the invariants – usually called integrity constraints in this context [9] – is much higher than the complexity of the programs themselves, the trigger approach has become very popular, but it can be shown that triggers cannot solve the problem in general [7].

*Massey University, Department of Information Systems, Private Bag 11222, Palmerston North, NZ, E-mail: [s.link|k.d.schewe]@massey.ac.nz

Another approach considers *greatest consistent specializations* (GCSs) [6, 8]. Here the goal is to replace a given program specification S and a given static invariant $\mathcal{I}$ by a slightly modified program specification $S_{\mathcal{I}}$ that is provably consistent with respect to $\mathcal{I}$. The modification should guarantee that "effects" of the original S are preserved within $S_{\mathcal{I}}$. For this the approach considers the specialization order on semantic equivalence classes of program specifications. The existing theory is based on infinitary logic $\mathcal{L}^{\omega}_{\infty\omega}$.

In order to shift the GCS approach from the purely theoretical framework ([6]) to an applicable theory we have to investigate computability of GCSs and decidability of preconditions that must be built. For these purposes it is preferable to obtain a tight connection with classical recursion theory [1]. Therefore, we will replace the underlying logic of [6] by first-order arithmetic logic. The paper will introduce a new theory of consistency enforcement based on this logic with almost all results from [6] carrying over in a modified form. On this basis, effectivity issues can be investigated for the first time.

We start in Section 2 with a brief review of arithmetic logic. Then we show the existence of predicate transformers with respect to this logic. In particular, relational program semantics becomes equivalent to predicate transformer semantics provided we guarantee the property of universal conjunctivity and the pairing condition. We even show in Section 3 that recursion theory can be extended to the arithmetic case, at least, if we are restricted to certain WHILE-loops.

With this background we can show that the GCS approach carries over to arithmetic logic. This will be done in Section 4. Many of the proofs in [6] only require slight changes. Computability cannot be guaranteed in general, since the building of least fixpoints requires to test for semantic equivalence, which is undecidable. For the case of FOR-loops, however, GCSs are computable. This will be shown in Section 5. Furthermore, we show how effective GCSs can be computed.

We argue that at least for one application field, i.e. databases as already mentioned, the restrictions are tolerable. For the general case some other pragmatic solutions must be applied [5]. We conclude with a short summary and outlook.

Due to the compact representations in this paper we recommend reading [3] for details.

2 Arithmetic Logic and Programming Semantics

Our study is based on first-order arithmetic logic [1, Ch.7], i.e. our logical language contains just the function symbols 0, s, $+$ and $*$ of arity 0, 1, 2 and 2. The informal meaning is as usual: the constant 0, the succesor function, addition and multiplication. By convenience $+$ and $*$ are written as infix operators. The only predicate symbol is the equality symbol $=$. Variables in our language will be $x_1, x_2, x_3, \dots$.

We use the notation $\mathbb{T}$ for the set of terms and $\mathbb{F}$ for the set of formulae. In addition, let V denote the set of variables. We allow all standard abbreviations including formulae *true* and *false*.

Semantically, we fix a structure with domain $\mathbb{N}$, the set of non-negative integers.

Then 0, s, $+$, $*$ and $=$ are interpreted in the usual way. For an interpretation it is then sufficient to consider a function $\sigma : V \to \mathbb{N}$. By the coincidence theorem it is even sufficient to be given the values $\sigma(x_i)$ for the free variables x_i in a term or a formula. In particular, we may always write σ as a k-tuple, if the number of free variables is k.

Finally, a k-ary relation $R \subseteq \mathbb{N}^k$ is called *arithmetical* iff it can be represented by a formula $Q \in \mathbb{F}$ in arithmetic logic (with free variables $x_1, \dots, x_k$), i.e. $(a_1, \dots, a_k) \in R$ holds iff $\models_\sigma Q$ holds for the interpretation defined by $\sigma(x_i) = a_i$ $(i = 1, \dots, k)$.

2.1 Predicate Transformers in Arithmetic Logic

In accordance with the existing theory on consistency enforcement in [6] each finite subset $X \subseteq V$ is called a *state space*. Each function $\sigma : X \to \mathbb{N}$ is called a *state* on X. Equivalently, a state is always representable by a k-tuple. For a fixed X let Σ $(= \Sigma(X))$ denote the *set of all states over* X.

A formula $\varphi \in \mathbb{F}$ with free variables $fr(\varphi)$ in X is then called an *X-formula* or an *invariant* on X. In order to emphasize the variables we sometimes write $\varphi(\vec{x})$ with a vector $\vec{x}$ of the state variables involved.

Then any pair of formulae $(\Delta(S), \Sigma_0(S))$ with $2k$ and k free variables, respectively, may be considered as defining the *relational semantics* of a program specification S. For convenience assume the first k free variables in $\Delta(S)$ to coincide with the free variables of $\Sigma_0(S)$.

According to our notation we sometimes write $\Delta(S)(\vec{x}, \vec{y})$ and $\Sigma_0(S)(\vec{x})$. So $\Delta(S)$ can be interpreted by state pairs, whereas $\Sigma_0(S)$ allows an interpretation by states. We interpret (σ, τ) with $\models_{(\sigma,\tau)} \Delta(S)$ as an *execution* of S with start state σ and a final state τ. Similarly, a state σ satisfying $\Sigma_0(S)$ is considered as a start state for S, in which a non-terminating execution of S exists.

Note that the model of relational semantics comprises daemonic non-determinism, non-termination and partial undefinedness.

In order to come to an axiomatic semantics based on the introduced logic of arithmetic, we associate with S two *predicate transformers* $wlp(S)$ and $wp(S)$ – i.e., functions from (equivalence classes) of formulae to (equivalence classes) of formulae – with the standard informal meaning:

- $wlp(S)(\varphi)$ characterizes those initial states σ such that each terminating execution of S starting in σ results in a state τ satisfying φ.
- $wp(S)(\varphi)$ characterizes those initial states σ such that each execution of S starting in σ terminates and results in a state τ satisfying φ.

The notation $wlp(S)(\varphi)$ and $wp(S)(\varphi)$ corresponds to the usual *weakest (liberal) precondition* of S with respect to the postcondition φ. In order to save space we shall often use the notation $w(l)p(S)(\varphi)$ to refer to both predicate transformers at a time. If this occurs in an equivalence, then omitting everything in parentheses gives the wp-part, whereas omitting just the parentheses results in the wlp-part.

From our introduction of $\Delta(S)$ and $\Sigma_0(S)$ the following definition is straightforward.

Definition 1 The *predicate transformers* associated with a program specification S on a state space X are defined as

$$\begin{array}{rcl} wlp(S)(\varphi(\vec{x})) & \Leftrightarrow & \forall \vec{y}.\Delta(S)(\vec{x},\vec{y}) \Rightarrow \varphi(\vec{y}) \qquad \text{and} \\ wp(S)(\varphi(\vec{x})) & \Leftrightarrow & (\forall \vec{y}.\Delta(S)(\vec{x},\vec{y}) \Rightarrow \varphi(\vec{y})) \wedge \neg\Sigma_0(S)(\vec{x}) \end{array}$$

for arbitrary X-formulae φ. □

The next step is to show that predicate transformers satisfying some nice conditions are sufficient for the definition of program specifications S. The conditions are the *pairing condition* and a slightly modified *universal conjunctivity* property. This gives the equivalence between the relational and the predicate transformer semantics.

We use the standard notation $w(l)p(S)^*(\varphi) \Leftrightarrow \neg w(l)p(S)(\neg\varphi)$ and refer to $wlp(S)^*$ and $wp(S)^*$ as the *dual predicate transformers*.

Proposition 1 *The predicate transformers $w(l)p(S)$ satisfy the following conditions:*

$$\begin{array}{rcl} wp(S)(\varphi) & \Leftrightarrow & wlp(S)(\varphi) \wedge wp(S)(true) \qquad \textit{and} \\ wlp(S)(\forall \vec{y}.Q(\vec{y}) \Rightarrow \varphi(\vec{x},\vec{y})) & \Leftrightarrow & \forall \vec{y}.Q(\vec{y}) \Rightarrow wlp(S)(\varphi(\vec{x},\vec{y})) \quad . \end{array}$$

Conversely, any pair of predicate transformers satisfying these two conditions defines $\Delta(S)(\vec{x},\vec{y}) \Leftrightarrow wlp(S)^*(\vec{x}=\vec{y})$ and $\Sigma_0(\vec{x}) \Leftrightarrow wp(S)^*(false)$.

Proof. We first show that $w(l)p(S)$ fulfil both conditions. Due to

$$\begin{array}{rcl} wp(S)(true) & \Leftrightarrow & (\forall \vec{y}.\Delta(S)(\vec{x},\vec{y}) \Rightarrow true) \wedge \neg\Sigma_0(S)(\vec{x}) \\ & \Leftrightarrow & \neg\Sigma_0(S)(\vec{x}) \end{array}$$

we receive the pairing condition

$$\begin{array}{rcl} wlp(S)(\varphi(\vec{x})) \wedge wp(S)(true) & \Leftrightarrow & (\forall \vec{y}.\Delta(S)(\vec{x},\vec{y}) \Rightarrow \varphi(\vec{y})) \wedge \neg\Sigma_0(S)(\vec{x}) \\ & \Leftrightarrow & wp(S)(\varphi(\vec{x})) \quad . \end{array}$$

The universal conjunctivity property follows from

$$\begin{array}{rcl} wlp(S)(\forall \vec{y}.Q(\vec{y}) \Rightarrow \varphi(\vec{x},\vec{y})) & \Leftrightarrow & \forall \vec{z}.\Delta(S)(\vec{x},\vec{z}) \Rightarrow \{\vec{x}/\vec{z}\}.(\forall \vec{y}.Q(\vec{y}) \Rightarrow \varphi(\vec{x},\vec{y})) \\ & \Leftrightarrow & \forall \vec{z}.\Delta(S)(\vec{x},\vec{z}) \Rightarrow (\forall \vec{y}.Q(\vec{y}) \Rightarrow \varphi(\vec{z},\vec{y})) \\ & \Leftrightarrow & \forall \vec{y}.\forall \vec{z}.(\Delta(S)(\vec{x},\vec{z}) \wedge Q(\vec{y}) \Rightarrow \varphi(\vec{z},\vec{y})) \\ & \Leftrightarrow & \forall \vec{y}.Q(\vec{y}) \Rightarrow \forall \vec{z}.(\Delta(S)(\vec{x},\vec{z}) \Rightarrow \varphi(\vec{z},\vec{y})) \\ & \Leftrightarrow & \forall \vec{y}.Q(\vec{y}) \Rightarrow \forall \vec{z}.(\Delta(S)(\vec{x},\vec{z}) \Rightarrow \{\vec{x}/\vec{z}\}.\varphi(\vec{x},\vec{y})) \\ & \Leftrightarrow & \forall \vec{y}.Q(\vec{y}) \Rightarrow wlp(S)(\varphi(\vec{x},\vec{y})) \end{array}$$

for the case that $\{\vec{y} \mid \models Q(\vec{y})\} \neq \emptyset$ holds. If this set is empty then $\neg Q(\vec{y})$ holds for all $\vec{y}$ and we have $wlp(S)(true) \Leftrightarrow true$ which is obviously valid.
Now, let f_{lp} and f_p be predicate transformers statisfying the pairing condition and the universal conjunctivity property. Then it remains to show $wlp(S) = f_{lp}(S)$ and $wp(S) = f_p(S)$. For an arbitrary X–formula φ we have

$$\models_\sigma \varphi(\vec{x}) \Leftrightarrow \models_\sigma \varphi'(\vec{x}) \quad \text{with} \quad \varphi'(\vec{x}) \Leftrightarrow \forall \vec{y}. (\vec{x} = \vec{y} \Rightarrow \varphi(\vec{y}))$$

Let σ be an arbitrary state with $\models_\sigma f_{lp}(S)(\varphi(\vec{x}))$. Then we compute

$$\begin{array}{lcl}
\models_\sigma f_{lp}(S)(\varphi(\vec{x})) & \Leftrightarrow & \models_\sigma f_{lp}(S)(\varphi'(\vec{x})) \\
& \Leftrightarrow & \models_\sigma f_{lp}(S)(\forall \vec{y}.\vec{x} = \vec{y} \Rightarrow \varphi(\vec{y})) \\
& \Leftrightarrow & \models_\sigma f_{lp}(S)(\forall \vec{y}.\neg\varphi(\vec{y}) \Rightarrow \vec{x} \neq \vec{y}) \\
& \Leftrightarrow & \models_\sigma \forall \vec{y}.\neg\varphi(\vec{y}) \Rightarrow f_{lp}(S)(\vec{x} \neq \vec{y}) \\
& \Leftrightarrow & \models_\sigma \forall \vec{y}.f_{lp}(S)^*(\vec{x} = \vec{y}) \Rightarrow \varphi(\vec{y}) \\
& \Leftrightarrow & \models_\sigma \forall \vec{y}.\Delta(S)(\vec{x},\vec{y}) \Rightarrow \varphi(\vec{y}) \\
& \Leftrightarrow & \models_\sigma wlp(S)(\varphi(\vec{x})) \quad ,
\end{array}$$

therefore the asserted equivalence. Furthermore, we have

$$\begin{array}{lcll}
wp(S)(\varphi) & \Leftrightarrow & wlp(S)(\varphi) \wedge wp(S)(true) & \text{(pairing condition)} \\
& \Leftrightarrow & wlp(S)(\varphi) \wedge \neg\Sigma_0(S)(\vec{x}) & \text{(Def. } wp(S)(true)\text{)} \\
& \Leftrightarrow & wlp(S)(\varphi) \wedge \neg f_p(S)^*(false) & \text{(Def. } \Sigma_0(S)\text{)} \\
& \Leftrightarrow & f_{lp}(S)(\varphi) \wedge \neg f_p(S)^*(false) & (wlp(S) = f_{lp}(S)) \\
& \Leftrightarrow & f_{lp}(S)(\varphi) \wedge f_p(S)(true) & \text{(Def. } f_p(S)^*\text{)} \\
& \Leftrightarrow & f_p(S)(\varphi) \quad , & \text{(pairing condition)}
\end{array}$$

which completes the proof. □

The next result gives a normal form representation of the predicate transformer $wlp(S)$, which will be useful in many proofs.

Lemma 1 *It is always possible to write $wlp(S)(\varphi)$ in the form*

$$wlp(S)(\varphi(\vec{x})) \quad \Leftrightarrow \quad \forall \vec{z}.wlp(S)^*(\vec{x} = \vec{z}) \Rightarrow \varphi(\vec{z})$$

Proof:
Obviously, we have $\varphi(\vec{x}) \Leftrightarrow \forall \vec{z}.\vec{x} = \vec{z} \Rightarrow \varphi(\vec{z}) \Leftrightarrow \forall \vec{z}.\neg\varphi(\vec{z}) \Rightarrow \vec{x} \neq \vec{z}$. Then the lemma follows immediately by applying the universal conjunctivity property. □

2.2 Guarded Commands

We now introduce the familiar language of guarded commands [4]. We use *skip*, *fail*, *loop* and parallel assignment $x_{i_1} := t_{i_1} \| \dots \| x_{i_k} := t_{i_k}$ with variables $x_{i_j} \in V$ and terms $t_{i_j} \in \mathbb{T}$ as basic commands. The informal meaning of the first three in this list is to change nothing, to be completely undefined and to do only non-terminating executions, respectively.

Complex commands are constructed from sequences $S_1; S_2$, choices $S_1 \Box S_2$, restricted choices $S_1 \boxtimes S_2$, unbounded choice $@x_j \bullet S$ and preconditioning $\mathcal{P} \to S$.

To define the semantics we simply have to define the predicate transformers. These are given as follows:

$$\begin{aligned}
w(l)p(skip)(\varphi) &\Leftrightarrow \varphi \\
w(l)p(fail)(\varphi) &\Leftrightarrow true \\
w(l)p(loop)(\varphi) &\Leftrightarrow false(\vee true) \\
w(l)p(x_{i_1} := t_{i_1} \| \dots \| x_{i_k} := t_{i_k})(\varphi) &\Leftrightarrow \{x_{i_1}/t_{i_1}, \dots, x_{i_k}/t_{i_k}\}.\varphi \\
w(l)p(S_1; S_2)(\varphi) &\Leftrightarrow w(l)p(S_1)(w(l)p(S_2)(\varphi)) \\
w(l)p(S_1 \Box S_2)(\varphi) &\Leftrightarrow w(l)p(S_1)(\varphi) \wedge w(l)p(S_2)(\varphi) \\
w(l)p(S_1 \boxtimes S_2)(\varphi) &\Leftrightarrow w(l)p(S_1)(\varphi) \wedge (wp(S_1)^*(true) \vee w(l)p(S_2)(\varphi)) \\
w(l)p(@x_j \bullet S)(\varphi) &\Leftrightarrow \forall x_j . w(l)p(S)(\varphi) \\
w(l)p(\mathcal{P} \to S)(\varphi) &\Leftrightarrow \mathcal{P} \Rightarrow w(l)p(S)(\varphi)
\end{aligned}$$

Here $\{x_{i_1}/t_{i_1}, \dots, x_{i_k}/t_{i_k}\}$ denotes the simultaneous substitution of the variables x_{i_j} by the terms t_{i_j}. We do not want to dispense with the *restricted choice*-operator $\boxtimes$ since it is needed to define IF S FI and DO S OD commands. For a deeper justification, please see [4]. Of course, we might always write $S_1 \Box wp(S_1)(false) \to S_2$ instead of $S_1 \boxtimes S_2$. However, this violates the orthogonality property of guarded commands which we want to maintain.

It is easy to verify the pairing condition and the universal conjunctivity property for these predicate transformers.

We say that S is an X-*command* for some state space X iff $w(l)p(S)(\varphi) \Leftrightarrow \varphi$ hold for each Y-formulae φ, where $X \cap Y = \emptyset$, and X is minimal with this property.

3 Recursion

In the last section we introduced the language of guarded commands together with an axiomatic semantics expressed via predicate transformers in arithmetic logic. So far, this language covers straightline non-deterministic partial programs extended by unbounded choice. We would like to go a bit further and investigate recursive programs expressed as least fixpoints $\mu T.f(T)$ with respect to a suitable order $\preceq$. This order will be the standard Nelson-order [4].

Unfortunately, we are not able to carry over the very general recursion theory from [4]. We have to restrict ourselves to simple WHILE-loops, i.e. $f(T) = \mathcal{P} \to S; T \Box \neg\mathcal{P} \to skip$, where the variable T does not occur within S. For convenience, we introduce command variables $T_1, T_2, \dots$. Throughout this section, we will use $f(T)$ to denote simple WHILE-loops as above.

3.1 The Nelson-Order

The idea of the Nelson-order is that whenever $S_1 \preceq S_2$ holds, then each terminating execution of S_1 is preserved within S_2, but a terminating execution in S_2 may be "approximated" in S_1 by a non-terminating execution. This leads to the following definition.

Definition 2 The *Nelson-order* is defined by

$$S_1 \preceq S_2 \Leftrightarrow (wlp(S_2)(\varphi) \Rightarrow wlp(S_1)(\varphi)) \wedge (wp(S_1)(\varphi) \Rightarrow wp(S_2)(\varphi))$$

for all φ. □

Particularly, we are interested in chains $\{f^i(loop)\}_{i \in \mathbb{N}}$ with respect to $\preceq$. Therefore, we define next a Gödel numbering g of guarded commands, which extends the Gödel numbering of terms and formulae from [1, p.327f.]. Let h denote this Gödel numbering for our logic. Recall the following definition:

$$h(0) = 1, \quad h(x_i) = 3^i, \quad h(s(t)) = 2 \cdot 3^{h(t)}, \quad h(t_1 + t_2) = 4 \cdot 3^{h(t_1)} \cdot 5^{h(t_2)},$$
$$h(t_1 * t_2) = 8 \cdot 3^{h(t_1)} \cdot 5^{h(t_2)}, \; h(t_1 = t_2) = 16 \cdot 3^{h(t_1)} \cdot 5^{h(t_2)}, \; h(\neg\varphi) = 32 \cdot 3^{h(\varphi)},$$
$$h(\varphi_1 \Rightarrow \varphi_2) = 64 \cdot 3^{h(\varphi_1)} \cdot 5^{h(\varphi_2)} \quad \text{and} \quad h(\forall x_i . \varphi) = 2^{6+i} \cdot 3^{h(\varphi)}.$$

In the same way we define

$$g(fail) = 1, \quad g(loop) = 2, \quad g(skip) = 4,$$
$$g(x_{i_1} := t_{i_1} \| \dots \| x_{i_k} := t_{i_k}) = 8 \cdot \prod_{j=1}^{k} prim(i_j)^{h(t_{i_j})},$$
$$g(S_1; S_2) = 16 \cdot 3^{g(S_1)} \cdot 5^{g(S_2)}, \quad g(S_1 \Box S_2) = 32 \cdot 3^{g(S_1)} \cdot 5^{g(S_2)},$$
$$g(S_1 \boxtimes S_2) = 64 \cdot 3^{g(S_1)} \cdot 5^{g(S_2)},$$
$$g(\mathcal{P} \rightarrow S) = 128 \cdot 3^{h(\mathcal{P})} \cdot 5^{g(S)}, \quad \text{and} \quad g(@x_j \bullet S) = 256 \cdot 3^j \cdot 5^{g(S)}$$

with the primitive recursive function *prim* taking n to the n'th prime number.

First we show that with this Gödel numbering g we may express all formulae $w(l)p(f^i(loop))(\varphi)$ by two arithmetic predicate transformers.

Lemma 2 *Let $f(T) = \mathcal{P} \rightarrow S; T \Box \neg\mathcal{P} \rightarrow skip$ such that T does not occur within S. Then for each $j \in \mathbb{N}$, there exist predicate transformers $\tau_l(j)$ and $\tau(j)$ on arithmetic predicates such that the following properties are satisfied:*

1. *for each arithmetic predicate $\varphi(\vec{x})$, the results of applying these predicate transformers are arithmetic predicates in i and x, say*

$$\chi_j^1(i, \vec{x}) = \tau_l(j)(\varphi(\vec{x})) \qquad \text{and} \qquad \chi_j^2(i, \vec{x}) = \tau(j)(\varphi(\vec{x}))$$

2. for $j = h(\varphi)$ we obtain

$$\begin{array}{ll} \forall \vec{x}.\forall i.\ \left(\chi_j^1(i,\vec{x}) \Leftrightarrow wlp(f^i(loop))(\varphi(\vec{x}))\right) & and \\ \forall \vec{x}.\forall i.\ \left(\chi_j^2(i,\vec{x}) \Leftrightarrow wp(f^i(loop))(\varphi(\vec{x}))\right) & \end{array}$$

with $\vec{x} = x_{i_1}, \dots, x_{i_k}$.

Proof. It is sufficient to prove the lemma for the case of S not containing loops itself. In general, program specifications can only have finitely many loops, so we can find the claimed predicate transformers $\tau_l(j)$ and $\tau(j)$ for the innermost loop first. Here, the involved program specification S, say S_0, is non-recursive. Having proven the lemma for this case, we obtain valid predicate transformers $wlp(S_1)$ and $wp(S_1)$ for the innermost loop S_1 by Lemma 3. Hence, without loss of generality we can assume that S in $f(T) = \mathcal{P} \rightarrow S; T \Box \neg\mathcal{P} \rightarrow skip$ is non-recursive.
For arbitrary program specifications T with $g(T) = i$ and arbitrary formulae $\varphi(\vec{x})$ with $h(\varphi) = j$ let us write $Q'_1(i,j,\vec{x}) = wlp(T)(\varphi(\vec{x}))$ and $Q'_2(i,j,\vec{x}) = wp(T)(\varphi(\vec{x}))$. If i, j are not Gödel numbers of programs or formulae, respectively, we may extend Q'_1 and Q'_2 arbitrarily. Let $prex(i,j)$ be the primitive recursive function that gives the exponent of the $j+1$-st prime number in the prime factorization of i. Then, we have

$$Q'_1(i,j,\vec{x}) = \begin{cases} true & , prex(i,0) = 0 \\ true & , prex(i,0) = 1 \\ h^{-1}(j) & , prex(i,0) = 2 \\ \{x_{i_1}/h^{-1}(j_1), \dots, x_{i_k}/h^{-1}(j_k)\}.h^{-1}(j) & , prex(i,0) = 3 \\ & prex(i,i_l) = j_l \\ & \text{with } 1 \le l \le k \\ Q'_1(prex(i,1), Q'_1(prex(i,2),j,\vec{x}),\vec{x}) & , prex(i,0) = 4 \\ Q'_1(prex(i,1),j,\vec{x}) \wedge Q'_1(prex(i,2),j,\vec{x}) & , prex(i,0) = 5 \\ Q'_1(prex(i,1),j,\vec{x}) \wedge (Q'_2(prex(i,1),7,\vec{x}) & \\ \quad \Rightarrow Q'_1(prex(i,2),j,\vec{x})) & , prex(i,0) = 6 \\ h^{-1}(prex(i,1)) \Rightarrow Q'_1(prex(i,2),j,\vec{x}) & , prex(i,0) = 7 \\ \forall x_{prex(i,1)}.Q'_1(prex(i,2),j,\vec{x}) & , prex(i,0) = 8 \end{cases}$$

We obtain a similar equation for $Q'_2(i,j,\vec{x})$ which does not depend on Q'_1. As this is a recursive definition, Q'_1 is not an arithmetic predicate. Note, however, that if we fix i and j, i.e., the program specification T and the formula φ, we can turn the equation into a formula of arithmetic logic.
Let us now consider just the case $T = f^k(loop)$ for our fixed mapping f on program specifications. For $k = 0$ we have $wlp(loop)(\varphi(\vec{x})) \Leftrightarrow true$. Furthermore, we get $wlp(f^{k+1}(loop))(\varphi(\vec{x})) \Leftrightarrow ((\mathcal{P} \Rightarrow wlp(S)(wlp(f^k(loop))(\varphi(\vec{x})))) \wedge (\neg\mathcal{P} \Rightarrow \varphi(\vec{x})))$. Thus, we may define a primitve recursive function $\bar{g}$ with $\bar{g}(0) = g(loop)$ and

$$\bar{g}(k+1) = g(f^{k+1}(loop)) = 32 \cdot 3^{128 \cdot 3^{h(\mathcal{P})} \cdot 5^{16 \cdot 3^{g(S)} \cdot 5^{\bar{g}(k)}}} \cdot 5^{128 \cdot 3^{h(\neg\mathcal{P})} \cdot 5^4}$$

such that

$$Q'_1(\bar{g}(k),j,\vec{x}) = wlp(f^k(loop))(\varphi(\vec{x}))$$

is satisfied. Now define an arithmetic formula $\bar{Q}(i,j,\vec{x})$ such that we have

$$\bar{Q}(h(\psi),h(\varphi),\vec{x}) \quad \Leftrightarrow \quad ((\mathcal{P} \Rightarrow wlp(S)(\psi)) \wedge (\neg\mathcal{P} \Rightarrow \varphi))$$

for arbitrary $\psi,\varphi \in \mathbb{F}$. As S is fixed and recursion-free we just take the right-hand side of the equivalence as the definition for $\bar{Q}(i,j,\vec{x})$ for Gödel numbers i,j of formulae and extend this to all i,j. If we take $Q_1(k,j,\vec{x}) = Q_1'(\bar{g}(k),j,\vec{x})$, we obtain (for $k > 0$)

$$Q_1(k,j,\vec{x}) \quad = \quad \bar{Q}(h(\psi),j,\vec{x})$$

with $\psi(\vec{x}) = wlp(f^{k-1}(loop))(\varphi(\vec{x}))$. Hence, also

$$\begin{array}{rcl} Q_1(0,j,\vec{x}) & = & true \quad \text{and} \\ Q_1(k+1,j,\vec{x}) & = & \bar{Q}(h(Q_1(k,j,\vec{x})),j,\vec{x}). \end{array}$$

Taking $\tau_l(j)(\varphi(\vec{x})) = \chi_j^1(k,\vec{x}) = Q_1(k,j,\vec{x})$ (for fixed j), this shows that $\chi_j^1(k,\vec{x})$ is arithmetic, as $\bar{Q}$ is arithmetic and arithmetic predicates are closed under primitive recursion. An analogous argument leads to arithmetic predicates $\chi_j^2(k,\vec{x}) = \tau(j)(\varphi(\vec{x}))$ for fixed j, thus proving the first part of the lemma. The equivalence in the second part follows immediately from the construction. □

With help of the arithmetic predicate transformers $\tau_l(j)$ and $\tau(j)$ from Lemma 2 we can now define a *limit operator* $S = \lim_{k\in\mathbb{N}} f^k(loop)$ via

$$\begin{array}{rcll} wlp(S)(\varphi(\vec{x})) & \Leftrightarrow & \forall k.\chi^1_{h(\varphi)}(k,\vec{x}) & \text{and} \\ wp(S)(\varphi(\vec{x})) & \Leftrightarrow & \exists k.\chi^2_{h(\varphi)}(k,\vec{x}) & . \end{array}$$

for $\chi^1_{h(\varphi)}(k,\vec{x}) = \tau_l(h(\varphi))(\varphi(\vec{x}))$ and $\chi^2_{h(\varphi)}(k,\vec{x}) = \tau(h(\varphi))(\varphi(\vec{x}))$.

Lemma 3 *The definition of $S = \lim_{i\in\mathbb{N}} f^i(loop)$ is sound.*

Proof. We first verify the universal conjunctivity property by direct calculation, namely

$$\begin{array}{rcl} wlp(S)(\forall\vec{z}.P(\vec{z}) \Rightarrow \varphi(\vec{x},\vec{z})) & \Leftrightarrow & \forall i.\chi^1_{h(\forall\vec{z}.P(\vec{z})\Rightarrow\varphi(\vec{x},\vec{z}))}(i,\vec{x}) \\ & \Leftrightarrow & \forall i.wlp(f^i(loop))(\forall\vec{z}.P(\vec{z}) \Rightarrow \varphi(\vec{x},\vec{z})) \\ & \Leftrightarrow & \forall i.\forall\vec{z}.P(\vec{z}) \Rightarrow wlp(f^i(loop))(\varphi(\vec{x},\vec{z}))) \\ & \Leftrightarrow & \forall\vec{z}.P(\vec{z}) \Rightarrow (\forall i.wlp(f^i(loop))(\varphi(\vec{x},\vec{z}))) \\ & \Leftrightarrow & \forall\vec{z}.P(\vec{z}) \Rightarrow \forall i.\chi^1_{h(\varphi)}(i,(\vec{x},\vec{z})) \\ & \Leftrightarrow & \forall\vec{z}.P(\vec{z}) \Rightarrow wlp(S)(\varphi(\vec{x},\vec{z})) \quad . \end{array}$$

For the second part of this Lemma, we first observe that

$$\begin{array}{rcl} wp(S)(\varphi(\vec{x})) & \Leftrightarrow & \exists i.\chi^2_{h(\varphi)}(i,\vec{x}) \\ & \Leftrightarrow & \exists i.wp(f^i(loop))(\varphi(\vec{x})) \\ & \Leftrightarrow & \exists i.wlp(f^i(loop))(\varphi(\vec{x})) \wedge wp(f^i(loop))(true) \end{array}$$

holds. In order to derive the pairing condition we verify both implications separately. Let us first show

$$wp(S)(\varphi) \quad \Rightarrow \quad wlp(S)(\varphi) \wedge wp(S)(true) \quad .$$

For a state σ with $\models_\sigma wp(S)(\varphi)$ it follows that $\models_\sigma wp(f^{i_0}(loop))(\varphi)$ holds, i.e. $\models_\sigma wlp(f^{i_0}(loop))(\varphi)$ and $\models_\sigma wp(f^{i_0}(loop))(true)$ for a particular $i_0 \in \mathbb{N}$. From $wp(S)(true) \Leftrightarrow \exists i.wp(f^i(loop))(true)$ we conclude $\models_\sigma wp(S)(true)$ and since $\{f^i(loop)\}_{i \in \mathbb{N}}$ is a chain it must be the case for every $i \in \mathbb{N}$ that either $f^i(loop) \preceq f^{i_0}(loop)$ or $f^{i_0}(loop) \preceq f^i(loop)$ holds which means either

$$\models_\sigma wlp(f^{i_0}(loop))(\varphi) \Rightarrow wlp(f^i(loop))(\varphi)$$

or

$$\models_\sigma wp(f^{i_0}(loop))(\varphi) \Rightarrow wlp(f^i(loop))(\varphi) \quad .$$

In every case, we have $\models_\sigma wlp(f^i(loop))(\varphi)$ for arbitrary $i \in \mathbb{N}$, therefore $\models_\sigma \forall i.wlp(f^i(loop))(\varphi)$, too and this is equivalent to $\models_\sigma wlp(S)(\varphi)$.
For the reverse direction

$$wlp(S)(\varphi) \wedge wp(S)(true) \quad \Rightarrow \quad wp(S)(\varphi)$$

we assume that $\models_\sigma \forall i.wlp(f^i(loop))(\varphi) \wedge \exists i.wp(f^i(loop))(true)$ holds. From this we derive $\models_\sigma wlp(f^{i_0}(loop))(\varphi) \wedge wp(f^{i_0}(loop))(true)$ for some $i_0 \in \mathbb{N}$, i.e. $\models_\sigma wp(f^{i_0}(loop))(\varphi)$ by the pairing condition of $f^{i_0}(loop)$. Finally, the assertion follows from $wp(S)(\varphi) \Leftrightarrow \exists i.wp(f^i(loop))(\varphi)$. □

3.2 Least Fixpoints

Now, we are going to show how to obtain the semantics for WHILE-loops. It is easy to see that the function $f(T) = \mathcal{P} \to S; T \Box \neg\mathcal{P} \to skip$ on guarded commands is monotonic in the Nelson order [4]. Then an immediate consequence of the last lemma is the existence of a least upper bound, which is just given by the limit operator.

Lemma 4 *The chain* $\{f^i(loop) \mid i \in \mathbb{N}\}$ *has a least upper bound, namely* $\lim_{i \in \mathbb{N}} f^i(loop)$. □

Proof. We have already seen in the proof of Lemma 3 that

$$wlp(\lim_{i \in \mathbb{N}} f^i(loop))(\varphi) \quad \Leftrightarrow \quad \forall i.wlp(f^i(loop))(\varphi)$$

holds which means we receive $wlp(\lim_{i \in \mathbb{N}} f^i(loop))(\varphi) \Rightarrow wlp(f^k(loop))(\varphi)$ for all $k \in \mathbb{N}$. In addition, we have obtained

$$wp(\lim_{i \in \mathbb{N}} f^i(loop))(\varphi) \quad \Leftrightarrow \quad \exists i.wp(f^i(loop))(\varphi)$$

and because of that $wp(f^k(loop))(\varphi) \Rightarrow wp(\lim_{i\in\mathbb{N}} f^i(loop))(\varphi)$ for all $k \in \mathbb{N}$. Consequently, $\lim_{i\in\mathbb{N}} f^i(loop)$ is an upper bound of the chain $\{f^i(loop) \mid i \in \mathbb{N}\}$ with respect to the Nelson-order.

Now, let T be an arbitrary upper bound of $\{f^i(loop) \mid i \in \mathbb{N}\}$. Then we have to show $\lim_{i\in\mathbb{N}} f^i(loop) \preceq T$ but this follows immediately from

$$wlp(T)(\varphi) \quad\Rightarrow\quad wlp(f^i(loop))(\varphi) \text{ for all } i \in \mathbb{N} \quad\Leftrightarrow\quad wlp(\lim_{i\in\mathbb{N}} f^i(loop))(\varphi)$$

and

$$wp(\lim_{i\in\mathbb{N}} f^i(loop))(\varphi) \quad\Leftrightarrow\quad wp(f^i(loop))(\varphi) \text{ for some } i \in \mathbb{N} \quad\Rightarrow\quad wp(T)(\varphi)\ .$$

Thus, $\lim_{i\in\mathbb{N}} f^i(loop)$ is the least upper bound as asserted. □

In the following we use the notation $\mu T.f(T)$ to denote the least fixpoint of f provided it exists. We now restrict ourselves to WHILE-loops.

Proposition 2 *Let $f(T) = \mathcal{P} \rightarrow S; T\Box\neg\mathcal{P} \rightarrow skip$. Then f has a least fixpoint with respect to $\preceq$, which is $\mu T.f(T) = \lim_{i\in\mathbb{N}} f^i(loop)$.*

Proof. First of all $\{f^i(loop) \mid i \in \mathbb{N}\}$ is a chain with respect to the Nelson-order since $loop$ is a minimum and f is monotonic. Therefore, $\bar{S} = \lim_{i\in\mathbb{N}} f^i(loop)$ is the least upper bound according to Lemma 4. At this point we want to verify that $\bar{S}$ is a fixpoint with respect to f. Due to

$$\begin{aligned}
wlp(f(\bar{S}))(\varphi) &\Leftrightarrow (\mathcal{P} \Rightarrow wlp(T)(wlp(\bar{S})(\varphi))) \wedge (\neg\mathcal{P} \Rightarrow \varphi)\\
&\Leftrightarrow (\mathcal{P} \Rightarrow wlp(T)(\forall i.i \in \mathbb{N} \Rightarrow wlp(f^i(loop))(\varphi))) \wedge (\neg\mathcal{P} \Rightarrow \varphi)\\
&\Leftrightarrow (\mathcal{P} \Rightarrow (\forall i.i \in \mathbb{N} \Rightarrow wlp(T)(wlp(f^i(loop))(\varphi)))) \wedge (\neg\mathcal{P} \Rightarrow \varphi)\\
&\Leftrightarrow (\forall i.i \in \mathbb{N} \Rightarrow (\mathcal{P} \Rightarrow wlp(T)(wlp(f^i(loop))(\varphi))) \wedge (\neg\mathcal{P} \Rightarrow \varphi)\\
&\Leftrightarrow \forall i.i \in \mathbb{N} \Rightarrow (\mathcal{P} \Rightarrow wlp(T)(wlp(f^i(loop))(\varphi)) \wedge (\neg\mathcal{P} \Rightarrow \varphi))\\
&\Leftrightarrow \forall i.i \in \mathbb{N} \Rightarrow wlp(\mathcal{P} \rightarrow T; f^i(loop)\Box\neg\mathcal{P} \rightarrow skip)(\varphi)\\
&\Leftrightarrow \forall i.i \in \mathbb{N} \Rightarrow wlp(f(f^i(loop)))(\varphi)\\
&\Leftrightarrow \forall i.i \in \mathbb{N} \Rightarrow wlp(f^i(loop))(\varphi)\\
&\Leftrightarrow wlp(\lim_{i\in\mathbb{N}} f^i(loop))(\varphi)\\
&\Leftrightarrow wlp(\bar{S})(\varphi)\quad ,
\end{aligned}$$

it remains to show $wp(f(\bar{S}))(true) \Leftrightarrow wp(\bar{S})(true)$. From the monotonicity of f it follows that $f(\bar{S})$ is a further upper bound of $\{f^i(loop) \mid i \in \mathbb{N}\}$ with respect to the Nelson-order, so we can conclude $\bar{S} \preceq f(\bar{S})$, especially

$$wp(\bar{S})(\varphi) \quad\Rightarrow\quad wp(f(\bar{S}))(\varphi)$$

Moreover, we receive

$$\begin{array}{lll} wp(f(\bar{S}))(\varphi) & \Leftrightarrow & (\mathcal{P} \Rightarrow wp(T)(wp(\bar{S})(\varphi))) \wedge (\neg\mathcal{P} \Rightarrow \varphi) \\ & \Leftrightarrow & (\mathcal{P} \Rightarrow wp(T)(\exists i.i \in \mathbb{N} \wedge wp(f^i(loop))(\varphi))) \wedge (\neg\mathcal{P} \Rightarrow \varphi) \\ & \Rightarrow & (\mathcal{P} \Rightarrow wp(T)(wp(f^i(loop))(\varphi))) \wedge (\neg\mathcal{P} \Rightarrow \varphi) \quad \text{for some } i \in \mathbb{N} \\ & \Leftrightarrow & wp(\mathcal{P} \rightarrow T; f^i(loop)\Box\neg\mathcal{P} \rightarrow skip)(\varphi) \quad \text{for some } i \in \mathbb{N} \\ & \Leftrightarrow & wp(\mathcal{P} \rightarrow T; f^i(loop)\Box\neg\mathcal{P} \rightarrow skip)(\varphi) \quad \text{for some } i \in \mathbb{N} \\ & \Leftrightarrow & wp(f^{i+1}(loop))(\varphi) \quad \text{for some } i \in \mathbb{N} \quad , \end{array}$$

i.e. as to be shown

$$wp(f(\bar{S}))(\varphi) \quad \Rightarrow \quad \exists i.i \in \mathbb{N} \wedge wp(f^i(loop))(\varphi) \quad \Leftrightarrow \quad wp(\bar{S})(\varphi) \quad .$$

Let $\bar{T}$ be an arbitrary fixpoint with respect to f. Since *loop* is a minimum with respect to the Nelson-order we have $loop \preceq \bar{T}$. Applying the monotonicity of f with respect to $\preceq$ again we obtain $f^n(loop) \preceq f^n(\bar{T}) = \bar{T}$ for arbitrary $n \in \mathbb{N}$, so $\bar{T}$ is an upper bound of $\{f^i(loop) \mid i \in \mathbb{N}\}$ with respect to the Nelson-order. But $\bar{S}$ is the least upper bound, thus $\bar{S} \preceq \bar{T}$ holds. □

Finally, in order to support also nested loops, we extend the Gödel numbering g to command variables and fixpoint expression letting

$$g(T_j) = 512 \cdot 3^j \qquad \text{and} \qquad g(\mu T_j.f(T_j)) = 1024 \cdot 3^j \cdot 5^{g(f(T_j))} \ .$$

For the extension of Q_1' and Q_2' from the proof of Lemma 2 we then need a function $\ell(x, j, k)$, which associates with the Gödel number $x = g(f(T_j))$ the Gödel number $g(f^j(loop))$. We omit the details.

4 Greatest Consistent Specializations

Now the foundations are laid to develop the theory of consistency enforcement on top of first-order arithmetic logic.

4.1 Consistency and Specialization

First we have to define consistency and the specialization preorder. This can be done in complete analogy to the case in [6].

Definition 3 Let $\mathcal{I}$ be an invariant on the state space X. Let S and T be commands on the state spaces Z and Y, respectively, with $Z \subseteq Y \subseteq X$.

- S is *consistent* with respect to $\mathcal{I}$ iff $\mathcal{I} \Rightarrow wlp(S)(\mathcal{I})$ holds.
- T *specializes* S (notation: $T \sqsubseteq S$) iff $w(l)p(S)(\varphi) \Rightarrow w(l)p(T)(\varphi)$ holds for all Z-formulae φ. □

Due to the pairing condition it is sufficient to consider only $\varphi = \mathit{true}$ for the *wp*-part in the specialization definition. The *wlp*-part can also be simplified in the known way. The proof of the next proposition is shifted into Appendix A. The result will play an important role in the proof of Theorem 2.

Proposition 3 *Let S and T be commands on the state spaces X and Y, respectively, with $X \subseteq Y$. Then $wlp(S)(\varphi) \Rightarrow wlp(T)(\varphi)$ holds for all X-formulae iff*

$$\{\vec{z}/\vec{x}\}.wlp(T')(wlp(S)^*(\vec{x} = \vec{z}))$$

holds, where $\vec{z}$ is a disjoint copy of $\vec{x}$ and T' results from T by renaming each x_i into z_i. □

Next we introduce the central notion for consistency enforcement, the GCS.

Definition 4 Let S be a Y-command and $\mathcal{I}$ an invariant on X with $Y \subseteq X$. The *greatest consistent specialization* (GCS) of S with respect to $\mathcal{I}$ is an X-command $S_{\mathcal{I}}$ with $S_{\mathcal{I}} \sqsubseteq S$, such that $S_{\mathcal{I}}$ is consistent with respect to $\mathcal{I}$ and each consistent specialization $T \sqsubseteq S$ satisfies $T \sqsubseteq S_{\mathcal{I}}$. □

First we show the existence of GCSs and their uniqueness up to semantic equivalence. Furthermore, GCSs with respect to conjunctions can be built successively. In both cases, the proofs from [8, 6] carry over without significant changes. Nevertheless, we will give the proofs in Appendix B.

Proposition 4 *The GCS $S_{\mathcal{I}}$ of S with respect to $\mathcal{I}$ always exists and is unique up to semantic equivalence. We can always write*

$$S_{\mathcal{I}} = (\mathcal{I} \rightarrow (S; @\vec{z}' \bullet \vec{z} := \vec{z}'; \mathcal{I} \rightarrow skip)) \boxtimes (\neg\mathcal{I} \rightarrow (S; @\vec{z}' \bullet \vec{z} := \vec{z}')) ,$$

where $\vec{z}$ refers to the free variables in $\mathcal{I}$ not occurring in S.

Furthermore, for two invariants $\mathcal{I}$ and $\mathcal{J}$ we always obtain that $\mathcal{I} \wedge \mathcal{J} \rightarrow S_{\mathcal{I}\wedge\mathcal{J}}$ and $\mathcal{I} \wedge \mathcal{J} \rightarrow (S_{\mathcal{I}})_{\mathcal{J}}$ are semantically equivalent. □

The normal form of $S_{\mathcal{I}}$ of Proposition 4 should be read as follows. Whenever $\mathcal{I}$ holds, we execute S and permit arbitrary assignments to state variables that are not affected by S. Subsequently, we test whether $\mathcal{I}$ was indeed invariant under the execution of S and these assignments. For the case that $\mathcal{I}$ does not hold, we do not need to check $\mathcal{I}$ again. Using the normal form of Proposition 4, we may derive $wp(S_{\mathcal{I}})(\mathit{true}) \Leftrightarrow wp(S)(\mathit{true})$ by direct computation. In fact, this is already obtainable from the definition of greatest consistent specializations. Anyway, this result allows us to concentrate on the predicate transformer $wlp(S)$.

4.2 An Upper Bound for GCSs

For practical applications the form of the GCS derived in Proposition 4 is almost worth nothing, since it involves testing the invariant after non-deterministic selection of arbitrary values. However, the form is useful in proofs.

A suitable form of the GCS should be built from GCSs of the basic commands involved in S. Let the result of such a naive syntactic replacement be denoted by $S'_{\mathcal{I}}$. In general, however, $S'_{\mathcal{I}}$ is not the GCS. It may not even be a specialization of S, or it may be a consistent specialization, but not the greatest one. An example for the latter case is $S = x := x - a; x := x + a$ with some constant $a \geq 1$ and $\mathcal{I} \equiv x \geq 1$.

We now formulate a technical condition which allows us to exclude this situation. Under this condition it will be possible to show that $S_{\mathcal{I}} \sqsubseteq S'_{\mathcal{I}}$ holds. The corresponding result will be called the *upper bound theorem.*

We need the notion of a *deterministic branch* S^+ of a command S, which requires $S^+ \sqsubseteq S$, $wp(S)^*(true) \Leftrightarrow wp(S^+)^*(true)$ and $wlp(S^+)^*(\varphi) \Rightarrow wp(S^+)(\varphi)$ to hold for all φ. Herein, the last condition expresses that S^+ is indeed deterministic, i.e., whenever $\models_{(\sigma,\tau)} \Delta(\vec{x},\vec{y})$ then $\models_\sigma \neg\Sigma_0(\vec{x})$ and whenever $\models_{(\sigma,\tau_1)} \Delta(\vec{x},\vec{y})$ and $\models_{(\sigma,\tau_2)} \Delta(\vec{x},\vec{y})$ hold then $\tau_1(\vec{x}) = \tau_2(\vec{x})$. Together, a deterministic branch S^+ of S is a deterministic specialization of S which comprises executions if and only if S does.

Furthermore, we need the notion of a *δ-constraint* for an X-command S. This is an invariant $\mathcal{J}$ on $X \cup X'$ with a disjoint copy X' of X, for which $\{\vec{x}'/\vec{x}\}.wlp(S')(\mathcal{J})$ holds, where S' results from S by renaming all x_i to x'_i. Thus, δ-constraints are exactly those formulae which are interpreted by state pairs and satisfied by a specification.

Finally, we write φ_σ for the characterizing formula of state σ.

Definition 5 Let $S = S_1; S_2$ be a Y-command such that S_i is a Y_i-command for $Y_i \subseteq Y$ $(i = 1, 2)$. Let $\mathcal{I}$ be some X-invariant with $Y \subseteq X$. Let $X - Y_1 = \{y_1, \dots, y_m\}$, $Y_1 = \{x_1, \dots, x_l\}$ and assume that $\{x'_1, \dots, x'_l\}$ is a disjoint copy of Y_1 disjoint also from X. Then S is in *δ-$\mathcal{I}$-reduced form* iff for each deterministic branch S_1^+ of S_1 the following two conditions – with $\vec{x} = (x_1, \dots, x_l)$, $\vec{x'} = (x'_1, \dots, x'_l)$ – hold:

- For all states σ with $\models_\sigma \neg\mathcal{I}$ we have, if $\varphi_\sigma \Rightarrow \{\vec{x}/\vec{x'}\}.(\forall y_1 \dots y_m.\mathcal{I})$ is a δ-constraint for S_1^+, then it is also a δ-constraint for $S_1^+ ; S_2$.
- For all states σ with $\models_\sigma \mathcal{I}$ we have, if $\varphi_\sigma \Rightarrow \{\vec{x}/\vec{x'}\}.(\forall y_1 \dots y_m.\neg\mathcal{I})$ is a δ-constraint for S_1^+, then it is also a δ-constraint for $S_1^+ ; S_2$. □

Informally, δ-$\mathcal{I}$-reducedness is a property of sequences $S_1; S_2$ which rules out occurences of interim states that wrongly cause an enforcement within any branch of S_1 but which is not relevant for the entire specification. If we for instance look again at the example above, then the GCS of $S = x := x - a; x := x + a$ with respect to $\mathcal{I} \equiv x \geq 1$ is certainly *skip*, but $(x := x - a)_{\mathcal{I}} = (x = 0 \vee x > a) \rightarrow x := x - a$. A simple replacement of basic commands by their respective GCSs leads in this case to $(x = 0 \vee x > a) \rightarrow x := x - a; x := x + a$ which is just a proper specialization of *skip*. The reason for this is, that S is not in $\mathcal{I}$-reduced form.
Arbitrary programs S are called $\mathcal{I}$-reduced iff all occurences of sequences within S are δ-$\mathcal{I}$-reduced.

Definition 6 Let S be an Y-command and $\mathcal{I}$ some X-invariant with $Y \subseteq X$. S is called *$\mathcal{I}$-reduced* iff the following holds:

- If S is one of $fail$, $skip$, $loop$ or an assignment, then S is always $\mathcal{I}$-reduced.
- If $S = S_1; S_2$, then S is $\mathcal{I}$-reduced iff S_1 and S_2 are $\mathcal{I}$-reduced and S is δ-$\mathcal{I}$-reduced.
- If S is one of $\mathcal{P} \to T$, $@y \bullet T$, $S_1 \Box S_2$ or $S_1 \boxtimes S_2$, then S is $\mathcal{I}$-reduced iff S_1 and S_2 or T respectively are $\mathcal{I}$-reduced.
- If $S = \mu T.f(T)$, then S is $\mathcal{I}$-reduced iff $f^n(loop)$ is $\mathcal{I}$-reduced for each $n \in \mathbb{N}$. $\Box$

With these technical preliminaries we may now state and prove the upper bound theorem. The proof itself is done by lengthy structural induction on guarded commands and therefore shifted to Appendix C.

Theorem 1 *Let $\mathcal{I}$ be an invariant on X and let S be some $\mathcal{I}$-reduced Y-command with $Y \subseteq X$. Let $S'_\mathcal{I}$ result from S as follows:*

- *Each restricted choice $S_1 \boxtimes S_2$ occurring within S will be replaced by $S_1 \Box$ $wlp(S_1)(false) \to S_2$.*
- *Then each basic command, i.e. skip, fail, loop and all assignments, will be replaced by their GCSs with respect to $\mathcal{I}$.*

Then $T \sqsubseteq S'_\mathcal{I}$ holds for each consistent specialization $T \sqsubseteq S$ with respect to $\mathcal{I}$. $\Box$

4.3 The General Form of a GCS

Theorem 1 has a flavour of compositionality, but it does not yet give the GCS. The idea of the main theorem on GCSs is to cut out from the upper bound $S'_\mathcal{I}$ those executions that are not allowed to occur in a specialization of S. This is accomplished by adding a precondition $\mathcal{P}$ whose meaning becomes obvious by Proposition 3. This leads to the following theorem.

Theorem 2 *Let $\mathcal{I}$, S and $S'_\mathcal{I}$ be as in Theorem 1. Let Z be a disjoint copy of the state space Y. With the formula*

$$\mathcal{P}(S,\mathcal{I},\vec{x}') \equiv \{\vec{z}/\vec{y}\}.wlp(S''_\mathcal{I}; \vec{z} = \vec{x}' \to skip)(wlp(S)^*(\vec{z} = \vec{y})) \quad ,$$

where $S''_\mathcal{I}$ results from $S'_\mathcal{I}$ by renaming the Y to Z, the GCS $S_\mathcal{I}$ is semantically equivalent to

$$@\vec{x}' \bullet \mathcal{P}(S,\mathcal{I},\vec{x}') \to (S'_\mathcal{I}; \vec{y} = \vec{x}' \to skip) \quad .$$

Proof. We take the form claimed in the theorem as a definition and verify the conditions in the definition of the GCS. If φ is an arbitrary Y-formula, we use the definition of dual predicate transformers to validate

$$wlp(S_{\mathcal{I}})^*(\varphi) \quad \Leftrightarrow \quad \exists \vec{x}'.\mathcal{P}(S,\mathcal{I},\vec{x}') \wedge wlp(S'_{\mathcal{I}})^*(\vec{y} = \vec{x}' \wedge \varphi) \quad .$$

If $\mathcal{P}(S,\mathcal{I},\vec{x}')$ holds, then

$$wlp(S'_{\mathcal{I}})^*(\vec{y} = \vec{x}' \wedge \varphi) \quad \Rightarrow \quad wlp(S)^*(\varphi)$$

is *true* for all Y-formulae φ by Proposition 3. But then it follows immediately that $wlp(S_{\mathcal{I}})^*(\varphi) \Rightarrow wlp(S)^*(\varphi)$ holds, hence $S_{\mathcal{I}} \sqsubseteq S$.
Consistency can be verified easily, since $S'_{\mathcal{I}}$ is already consistent with respect to $\mathcal{I}$, namely

$$\begin{aligned}
\mathcal{I} \quad & \Rightarrow \quad wlp(S'_{\mathcal{I}})(\mathcal{I}) \\
& \Rightarrow \quad wlp(S'_{\mathcal{I}})(\vec{y} = \vec{x}' \Rightarrow wlp(skip)(\mathcal{I})) \\
& \Leftrightarrow \quad wlp(S'_{\mathcal{I}})(wlp(\vec{y} = \vec{x}' \rightarrow skip)(\mathcal{I})) \\
& \Leftrightarrow \quad wlp(S'_{\mathcal{I}}; \vec{y} = \vec{x}' \rightarrow skip)(\mathcal{I}) \\
& \Rightarrow \quad \forall \vec{x}'.\mathcal{P}(S,\mathcal{I},\vec{x}') \Rightarrow wlp(S'_{\mathcal{I}}; \vec{y} = \vec{x}' \rightarrow skip)(\mathcal{I}) \\
& \Leftrightarrow \quad wlp(@\vec{x}' \bullet \mathcal{P}(S,\mathcal{I},\vec{x}') \rightarrow S'_{\mathcal{I}}; \vec{y} = \vec{x}' \rightarrow skip)(\mathcal{I}) \\
& \Leftrightarrow \quad wlp(S_{\mathcal{I}})(\mathcal{I}) \quad .
\end{aligned}$$

Therefore we have the consistency of $S_{\mathcal{I}}$ with respect to $\mathcal{I}$. Note, that the second implication in the computation above holds due to the monotonicity of $wlp(S'_{\mathcal{I}})$ applied to $\mathcal{I} \Rightarrow (\vec{y} = \vec{x}' \Rightarrow \mathcal{I})$.
Finally, let T be an arbitrary consistent specialization of S. We assume without loss in generality that $wp(T)(true) \Leftrightarrow true$ holds. From Theorem 1 we already get $T \sqsubseteq S'_{\mathcal{I}}$. From this we compute

$$\begin{aligned}
w(l)p(\underbrace{S'_{\mathcal{I}}; \vec{y} = \vec{x}' \rightarrow skip}_{S_{\mathcal{I}}^{\vec{x}'}})(\varphi) \quad & \Leftrightarrow \quad w(l)p(S'_{\mathcal{I}})(w(l)p(\vec{y} = \vec{x}' \rightarrow skip)(\varphi)) \\
& \Rightarrow \quad w(l)p(T)(w(l)p(\vec{y} = \vec{x}' \rightarrow skip)(\varphi)) \\
& \Leftrightarrow \quad w(l)p(\underbrace{T; \vec{y} = \vec{x}' \rightarrow skip}_{T^{\vec{x}'}})(\varphi) \quad ,
\end{aligned}$$

i.e. $T^{\vec{x}'} \sqsubseteq S_{\mathcal{I}}^{\vec{x}'}$. At this point it suffices to show $wp(T^{\vec{x}'})^*(true) \Rightarrow \mathcal{P}(S,\mathcal{I},\vec{x}')$, because

$$\begin{aligned}
w(l)p(\mathcal{P}(S,\mathcal{I},\vec{x}') \rightarrow S_{\mathcal{I}}^{\vec{x}'})(\varphi) \quad & \Leftrightarrow \quad \mathcal{P}(S,\mathcal{I},\vec{x}') \Rightarrow w(l)p(S_{\mathcal{I}}^{\vec{x}'})(\varphi) \\
& \Rightarrow \quad wp(T^{\vec{x}'})^*(true) \Rightarrow w(l)p(S_{\mathcal{I}}^{\vec{x}'})(\varphi) \\
& \Rightarrow \quad wp(T^{\vec{x}'})^*(true) \Rightarrow w(l)p(T^{\vec{x}'})(\varphi) \\
& \Leftrightarrow \quad w(l)p(\underbrace{wp(T^{\vec{x}'})^*(true) \rightarrow T^{\vec{x}'}}_{T^{\vec{x}'}})(\varphi)
\end{aligned}$$

implies immediately $T^{\vec{x}'} \sqsubseteq \mathcal{P}(S,\mathcal{I},\vec{x}') \to S_{\mathcal{I}}^{\vec{x}'}$ and we obtain $\forall \vec{x}' \bullet T^{\vec{x}'} \sqsubseteq \forall \vec{x}' \bullet \mathcal{P}(S,\mathcal{I},\vec{x}') \to S_{\mathcal{I}}^{\vec{x}'}$, consequently. The formula on the left-hand side is equivalent to T, whereas the one on the right-hand side is equivalent to $S_{\mathcal{I}}$.
Assume there is a state $\vec{a}$, in which $\mathcal{P}(S,\mathcal{I},\vec{x}')$ does not hold. From Proposition 3 we get the existence of a state $\vec{b}$ with

$$\models_{\vec{a}} \neg \left(wlp(S)(\vec{y} \neq \vec{b}) \Rightarrow wlp(S'_{\mathcal{I}}; \vec{y} = \vec{x}' \to skip)(\vec{y} \neq \vec{b}) \right) \quad ,$$

which is equivalent to

$$\models_{\vec{a}} wlp(S)(\vec{y} \neq \vec{b}) \wedge \neg wlp(S'_{\mathcal{I}})(\vec{y} = \vec{x}' \Rightarrow \vec{y} \neq \vec{b})$$

and this, finally, to

$$\models_{\vec{a}} wlp(S)(\vec{y} \neq \vec{b}) \wedge wlp(S'_{\mathcal{I}})^*(\vec{y} = \vec{x}' \wedge \vec{y} = \vec{b}) \quad .$$

Hence $\vec{x}' = \vec{b}$ must hold by definition of characterizing state formulae. On the other hand we receive $\models_{\vec{a}} wlp(T)(\vec{y} \neq \vec{b})$ due to $T \sqsubseteq S$ and together with

$$\begin{array}{lcl} wlp(T^{\vec{x}'})(false) & \Leftrightarrow & wlp(T)(\vec{y} = \vec{x}' \Rightarrow false) \\ & \Leftrightarrow & wlp(T)(\vec{y} \neq \vec{x}') \\ & \Leftrightarrow & wlp(T)(\vec{y} \neq \vec{b}) \end{array}$$

we conclude $\models_{\vec{a}} wlp(T^{\vec{x}'})(false)$. From the pairing condition $wp(T^{\vec{x}'})(false) \Leftrightarrow wlp(T^{\vec{x}'})(false) \wedge wp(T^{\vec{x}'})(true)$ and

$$wp(T^{\vec{x}'})(true) \Leftrightarrow wp(T)(\vec{y} = \vec{x}' \Rightarrow true) \Leftrightarrow wp(T)(true) \Leftrightarrow true$$

follows $\models_{\vec{a}} wp(T^{\vec{x}'})(false)$, which is equivalent to $\models_{\vec{a}} \neg wp(T^{\vec{x}'})^*(true)$. □

Note that if we consider deterministic branches as a pragmatic approach suggested in [6], then the unbounded choice in Theorem 2 disappears. We omit further details.

The charaterization of GCSs according to Theorem 2 makes it formally possible to reduce consistency enforcement to a simple syntactical replacement (the forming of $S'_{\mathcal{I}}$) and to an investigation of a guard, namely $\mathcal{P}(S,\mathcal{I},\vec{x}')$.

5 Computability and Decidability

We have now reached the stage, where we can say that the GCS approach could have been succesfully developed with respect to arithmetic logic. Thus, we can turn to the original intention of this paper: computability and decidability issues.

Taking the general form of the GCS in Theorem 2 we may now ask, whether we can find an algorithm to compute the GCS. We may further ask, whether the result is effective. In general it will not be possible to compute the GCS, but we will identify subcases, for which effective GCSs can be computed.

5.1 The Computability of GCSs

First consider the computability problem. Taking our Gödel numberings h for terms and formulae and g for commands, we have already exploited their inversibility. From this we obtain the following immediate consequence.

Lemma 5 *For each $n \in \mathbb{N}$ it is decidable, whether n is the Gödel number of a term, a formula or a guarded command.* □

Next we consider the upper bound $S'_{\mathcal{I}}$ that occurs in the GCS. Since this is only a syntactic transformation, we may now conclude that $(S, \mathcal{I}) \mapsto S'_{\mathcal{I}}$ is computable. Hence it is sufficient to investigate the computability for the precondition $\mathcal{P}(S, \mathcal{I}, \vec{x}')$ for arbitrary $\vec{x}'$.

These conditions involve the predicate transformers $wlp(S)$ and $wlp(S'_{\mathcal{I}})$. According to our definition of axiomatic semantics for commands, we know that building these predicate transformers is simple done by syntactic replacement operations. By exploiting our Gödel numbering h again, we conclude that for recursion-free S the mapping

$$(S, \mathcal{I}, \vec{x}') \mapsto \mathcal{P}(S, \mathcal{I}, \vec{x}')$$

– and hence $(S, \mathcal{I}) \mapsto S_{\mathcal{I}}$, too – is computable.
However, if S involves a loop, then $S'_{\mathcal{I}}$ also involves a loop. In order to determine $wlp(S)$ and $wlp(S'_{\mathcal{I}})$ we have to use the limit operator. For a loop $\mu T_j.f(T_j)$ this means to build $wlp(f^i(loop))$ for all $i \in \mathbb{N}$. This is only possible, if there is some $n \in \mathbb{N}$ such that $wlp(f^n(loop)) = wlp(f^m(loop))$ holds for all $m \geq n, m \in \mathbb{N}$. This means that we have a bounded loop (or equivalently a FOR-loop).

Proposition 5 *If recursive guarded commands are restricted to bounded loops, then GCSs are computable, i.e. the function $(S, \mathcal{I}) \mapsto S_{\mathcal{I}}$ is computable. In general, however, the GCS cannot be computed.* □

5.2 Effective GCSs

Even, if the GCS $S_{\mathcal{I}}$ can be computed from a given command S and the invariant $\mathcal{I}$, the result still contains the preconditions $\mathcal{P}(S, \mathcal{I}, \vec{x}')$. If such a precondition is undecidable, then the GCSs will not be effective. We will demonstrate how effective GCSs can be computed.

Therefore, we consider the proof of the upper bound theorem (see Appendix C) again. The next result shows that we have already proven more than we needed.

Lemma 6 *Let T be a program specification on Y and $\mathcal{I}$ a static constraint on X with $Y \subseteq X$.*

1. *If $T = P \rightarrow S$, then $T_{\mathcal{I}} = P \rightarrow S_{\mathcal{I}}$.*
2. *If $T = S_1 \square S_2$, then $T_{\mathcal{I}} = (S_1)_{\mathcal{I}} \square (S_2)_{\mathcal{I}}$.*
3. *If $T = @y \bullet S$, then $T_{\mathcal{I}} = @y \bullet S_{\mathcal{I}}$.*

Proof. The Propositions 7, 8 and 9 show the specialization in one direction. For the reverse specialization, one shows straightforwardly that $P \rightarrow S_{\mathcal{I}}$, $(S_1)_{\mathcal{I}} \Box (S_2)_{\mathcal{I}}$ and $@y \bullet S_{\mathcal{I}}$ are $\mathcal{I}$-consistent specializations of $P \rightarrow S$, $S_1 \Box S_2$ and $@y \bullet S$, respectively.
□

Note, that Lemma 6 does not hold for the case of sequences, even if they are δ-$\mathcal{I}$-reduced. Although Proposition 11 gives us of course specialization in one direction, the reverse specialization does not hold in general. The reason why $(S_1)_{\mathcal{I}}; (S_2)_{\mathcal{I}}$ is not a specialization of $S_1; S_2$ is that $wlp(S_2)(\varphi)$ is not necessarily a state formula of the underlying S_1 state space.

The next lemma will give us a computation of effective GCSs for program specifications S that only use basic commands, choices, guards and sequences. We dispense with the case of restricted choices.

Lemma 7 *Let S be a program specification on X built of basic commands, choices, guards with decidable preconditions and sequences. If φ is a decidable state formula on X, then $wlp(S)(\varphi)$ and $wlp(S)^*(\varphi)$ are decidable as well.*

Proof. The proof is a straightforward structural induction that makes use of the closure properties for decidable arithmetical predicates. □

It it well-known that every first-order predicate formula φ is equivalent to a formula $Q_1 x_1 \dots Q_k x_k . \psi$ where $Q_i \in \{\forall, \exists\}$ for $i = 1, \dots, k$ and ψ is quantifier-free. This result carries immediately over to guarded commands with respect to the @-operator.

Lemma 8 *Each guarded command S, whose occurences of loops are all bounded, can be written in the form $@x_1 \bullet \dots @x_n \bullet S'$ such that S' does not contain an unbounded choice operator @.*

Proof. The only interesting case is the one for bounded loops. Applying the predicate transformer wlp here results in a finite conjunction, whereas wp gives a finite disjunction. □

Let us all bring together and consider a program specification S for which all occurences of loops are bounded and all preconditions are decidable. In a first step, we replace all occurences of the restricted choice operator ⊠ in the usual way. Then we apply Lemma 8 that provides us with a specification $T = @y_1 \bullet \dots @y_n \bullet R$ that is semantically equivalent to S. Lemma 6 tells us then not to worry about the occurences of unbounded-choice operators, i.e., $T_{\mathcal{I}} = @y_1 \bullet \dots @y_n \bullet R_{\mathcal{I}}$. We apply the main theorem (Theorem 2) to compute $R_{\mathcal{I}}$ and conclude by Lemma 7 that all preconditions of the form $\mathcal{P}(S', \mathcal{I}, \vec{x}')$ are decidable. Finally, we obtain the following result.

Proposition 6 *Let S be a program specification such that every loop is bounded and all preconditions are decidable. Let $\mathcal{I}$ be a decidable static constraint. Then we can compute the GCS $S_{\mathcal{I}}$ in the form $S_{\mathcal{I}} = @y_1 \bullet \dots @y_n \bullet T_{\mathcal{I}}$, where $T_{\mathcal{I}}$ has the form of Theorem 2 with all preconditions $\mathcal{P}(T', \mathcal{I}, \vec{x}')$ being decidable.* □

6 Conclusion

In this article we considered the GCS approach to consistency enforcement presented in [6]. We could show that the underlying theory of predicate transformers could be carried over from an infinitary logic to first-order arithmetic logic. We were even able to do this for recursive program specifications by exploiting Gödel numberings for terms, formulae and guarded commands. However, the used recursive program specifications are slightly restricted with respect to the more general theory in [4].

Then we could show that the existence and uniqueness of GCSs, the commutativity result from [8] and the fundamental compositionality result carry over to the new logic. This allows to study computability and decidability issues. We could show that the GCS is computable for program specifications where all loops are bounded. Moreover, effective GCSs can be computed when preconditions within guards and the given static constraint are decidable.

There are at least three more problems we would like to approach next. Firstly, we would like to study the Goldfarb classification [2] and its impact to GCS construction. More precisely, we look for a characterization of those static invariants $\mathcal{I}$ for which $\mathcal{I}$-reducedness is decidable. Secondly, we would like to look at weakened approaches to consistency enforcement, e.g. the one presented in [5] and to discuss computability and decidability for this approach as well. Thirdly and finally, we would like to address the problems of GCSs – and weakened approaches – with respect to basic commands. In particular, it would be nice to see how GCSs for various classes of relational constraints would look like.

A Appendix A: Proof of the Normal Form for Specialization

Proposition 3. *Let S and T be commands on the state spaces X and Y, respectively, with $X \subseteq Y$. Then $wlp(S)(\varphi) \Rightarrow wlp(T)(\varphi)$ holds for all X-formulae iff*

$$\{\vec{z}/\vec{x}\}.wlp(T')(wlp(S)^*(\vec{x} = \vec{z}))$$

holds, where $\vec{z}$ is a disjoint copy of $\vec{x}$ and T' results from T by renaming each x_i into z_i.

Proof. The normal form representation from Lemma 1 gives for $wlp(T')$ the equivalence from $wlp(T')(wlp(S)^*(\vec{x} = \vec{z}))$ to

$$\forall \vec{z'}.wlp(T')^*(\vec{z} = \vec{z'}) \Rightarrow \{\vec{z}/\vec{z'}\}.wlp(S)^*(\vec{x} = \vec{z}).$$

Now, S is defined on X which results in

$$\{\vec{z}/\vec{z'}\}.wlp(S)^*(\vec{x} = \vec{z}) \quad \Leftrightarrow \quad wlp(S)^*(\vec{x} = \vec{z'}) \quad .$$

Hence, it is sufficient to show the equivalence between

1. $wlp(S)(\varphi) \quad \Rightarrow \quad wlp(T)(\varphi)$ for all X-formulae φ and
2. $\{\vec{z}/\vec{x}\}.(\forall \vec{z}'.wlp(T')^*(\vec{z} = \vec{z}') \Rightarrow wlp(S)^*(\vec{x} = \vec{z}'))$.

Let us assume that (1) holds. By renaming, $wlp(S')(\varphi) \Rightarrow wlp(T')(\varphi)$ holds for all Z-formulae φ. In particular, if $\varphi \equiv \vec{z} = \vec{a}$ for some state $\vec{a}$, then $wlp(S')(\vec{z} = \vec{a}) \Leftrightarrow \{\vec{x}/\vec{z}\}.wlp(S)^*(\vec{x} = \vec{a})$. But then,

$$\forall \vec{z}'.(wlp(T')^*(\vec{z} = \vec{z}') \Rightarrow \{\vec{x}/\vec{z}\}.wlp(S)^*(\vec{x} = \vec{z}'))$$

must be valid and this implies (2).
Suppose that (2) holds. Again, Lemma 1 can be employed to show the equivalence of $wlp(T)^*(\varphi)$ with arbitrary X-formula φ to

$$\exists \vec{z}'.(\{\vec{z}/\vec{x}\}.wlp(T')^*(\vec{z} = \vec{z}') \wedge \varphi(\vec{z}')) \quad .$$

With $\forall \vec{z}'.(wlp(T')^*(\vec{z} = \vec{z}') \Rightarrow \{\vec{x}/\vec{z}'\}.wlp(S)^*(\vec{x} = \vec{z}'))$ follows immediately

$$\{\vec{z}/\vec{x}\}.(\exists \vec{z}'.(wlp(S)^*(\vec{x} = \vec{z}') \wedge \varphi(\vec{z}'))) \quad ,$$

which is equivalent to $wlp(S)^*(\varphi)$ by Lemma 1. This gives the proof. □

B Appendix B: Existence, Normal Form Representation and Commutativity of GCSs

In the appendix we give a detailed proof of Proposition 4.
Proposition 4. *The GCS $S_{\mathcal{I}}$ of S with respect to $\mathcal{I}$ always exists and is unique up to semantic equivalence. We can always write*

$$S_{\mathcal{I}} = (\mathcal{I} \rightarrow (S; @\vec{z}' \bullet \vec{z} := \vec{z}'; \mathcal{I} \rightarrow skip)) \boxtimes (\neg\mathcal{I} \rightarrow (S; @\vec{z}' \bullet \vec{z} := \vec{z}')) \ ,$$

where $\vec{z}$ refers to the free variables in $\mathcal{I}$ not occurring in S.

Furthermore, for two invariants $\mathcal{I}$ and $\mathcal{J}$ we always obtain that $\mathcal{I} \wedge \mathcal{J} \rightarrow S_{\mathcal{I}\wedge\mathcal{J}}$ and $\mathcal{I} \wedge \mathcal{J} \rightarrow (S_{\mathcal{I}})_{\mathcal{J}}$ are semantically equivalent.
Proof. First we show the existence and uniqueness up to semantic equivalence of GCS. We set

$$\mathcal{T} = \{T \mid T \sqsubseteq S \text{ and } T \text{ is consistent with respect to } \mathcal{I}\} \quad .$$

If the least upper bound $S_{\mathcal{I}}$ of $\mathcal{T}$ with respect to the specialization $\sqsubseteq$ exists, then this must be the GCS. Therefore, we have the uniqueness up to semantic equivalence. We now verify the conditions from Definition 4 for the program specification $S_{\mathcal{I}}$ above. Let φ be an arbitrary state formula on Y. Then we receive

$$\begin{aligned}
wlp(S_{\mathcal{I}})^*(\varphi) \quad &\Leftrightarrow \quad (\mathcal{I} \wedge wlp(S)^*(\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.(\mathcal{I} \wedge \varphi))) \vee \\
&\qquad (\neg\mathcal{I} \wedge wlp(S)^*(\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.\varphi)) \\
&\Leftrightarrow \quad (\mathcal{I} \wedge wlp(S)^*((\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.\mathcal{I}) \wedge \varphi)) \vee (\neg\mathcal{I} \wedge wlp(S)^*(\varphi)) \\
&\Rightarrow \quad (\mathcal{I} \wedge wlp(S)^*(\varphi)) \vee (\neg\mathcal{I} \wedge wlp(S)^*(\varphi)) \\
&\Leftrightarrow \quad wlp(S)^*(\varphi) \quad .
\end{aligned}$$

Doing this we have made use of the dual predicate transformers' monotonicity property and the fact that variables z_i do not occur within φ. Then the asserted specialization $S_\mathcal{I} \sqsubseteq S$ follows from the same computation for *wp* instead of *wlp*. Next we consider

$$\begin{array}{rcl} wlp(S_\mathcal{I})(\mathcal{I}) & \Leftrightarrow & (\mathcal{I} \Rightarrow wlp(S)(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.(\mathcal{I} \Rightarrow \mathcal{I}))) \wedge \\ & & (\neg\mathcal{I} \Rightarrow wlp(S)(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\mathcal{I})) \\ & \Leftrightarrow & \neg\mathcal{I} \Rightarrow wlp(S)(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\mathcal{I}) \\ & \Leftrightarrow & \mathcal{I} \vee \neg wlp(S)(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\mathcal{I}) \quad . \end{array}$$

and obtain $\mathcal{I} \Rightarrow wlp(S_\mathcal{I})(\mathcal{I})$ which means that the above $S_\mathcal{I}$ is indeed consistent with respect to $\mathcal{I}$.
Let $\vec{x} = \vec{y}$ be a characterizing state formula and $T \sqsubseteq S$ an arbitrary, but $\mathcal{I}$-consistent specialization of S. Then we ditinguish two cases.
Case 1. We assume $\vec{x} = \vec{y} \Rightarrow \neg\mathcal{I}$ and therefore we conclude $wlp(T)^*(\vec{x} = \vec{y}) \Rightarrow wlp(T)^*(\neg\mathcal{I}) \Rightarrow \neg\mathcal{I}$ using the monotonicity of $wlp(S)^*$ and consistency of T. Moreover, it follows

$$\begin{array}{rcl} wlp(T)^*(\vec{x} = \vec{y}) & \Rightarrow & \neg\mathcal{I} \wedge wlp(S)^*(\vec{x} = \vec{y}) \\ & \Rightarrow & \neg\mathcal{I} \wedge wlp(S)^*(\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.\vec{x} = \vec{y}) \\ & \Rightarrow & wlp(S_\mathcal{I})^*(\vec{x} = \vec{y}) \quad . \end{array}$$

For the first implication we simply use the specialization $T \sqsubseteq S$, for the second we refer to the monotonicity applied to $\vec{x} = \vec{y} \Rightarrow \exists \vec{z}'.\{\vec{z}/\vec{z}'\}.\vec{x} = \vec{y}$ and the last one follows from the first line of the computation of $wlp(S_\mathcal{I})^*$
Case 2. Starting from $\vec{x} = \vec{y} \Rightarrow \mathcal{I}$ gives $wlp(T)^*(\vec{x} = \vec{y}) \Leftrightarrow wlp(T)^*(\mathcal{I} \wedge \vec{x} = \vec{y})$, subsequentely. We compute the following using $T \sqsubseteq S$ and the monotonicity of $wlp(S)^*$

$$\begin{array}{rcl} wlp(T)^*(\vec{x} = \vec{y}) & \Rightarrow & wlp(S)^*(\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.(\mathcal{I} \wedge \vec{x} = \vec{y})) \wedge \\ & & wlp(S)^*(\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.\vec{x} = \vec{y}) \\ & \Rightarrow & (\mathcal{I} \wedge wlp(S)^*(\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.(\mathcal{I} \wedge \vec{x} = \vec{y}))) \vee \\ & & (\neg\mathcal{I} \wedge wlp(S)^*(\exists \vec{z}'.\{\vec{z}/\vec{z}'\}.\vec{x} = \vec{y})) \\ & \Leftrightarrow & wlp(S_\mathcal{I})^*(\vec{x} = \vec{y}) \quad . \end{array}$$

This first step has brought us to $wlp(T)^*(\vec{x} = \vec{y}) \Rightarrow wlp(S_\mathcal{I})^*(\vec{x} = \vec{y})$, i.e. $wlp(S_\mathcal{I})(\vec{x} \neq \vec{y}) \Rightarrow wlp(T)(\vec{x} \neq \vec{y})$. For arbitrary state formula φ we have $\varphi(\vec{x}) \Leftrightarrow \forall \vec{y}.\neg\varphi(\vec{y}) \Rightarrow \vec{x} \neq \vec{y}$ and therefore

$$\begin{array}{rcl} wlp(S_\mathcal{I})(\varphi(\vec{x})) & \Leftrightarrow & \forall \vec{y}.\neg\varphi(\vec{y}) \Rightarrow wlp(S_\mathcal{I})(\vec{x} \neq \vec{y}) \\ & \Rightarrow & \forall \vec{y}.\neg\varphi(\vec{y}) \Rightarrow wlp(T)(\vec{x} \neq \vec{y}) \\ & \Leftrightarrow & wlp(T)(\varphi(\vec{x})) \quad , \end{array}$$

using the universal conjunctivity property of *wlp*. Thus, we obtain $wlp(T)^*(\varphi) \Rightarrow wlp(S_\mathcal{I})^*(\varphi)$ for all φ. On top of that $wp(T)^*(false) \Rightarrow wp(S)^*(false) \Rightarrow$

$wp(S_{\mathcal{I}})^*(false)$ holds as well, due to the specialization $T \sqsubseteq S$ and the first line of the computation of $wlp(S_{\mathcal{I}})^*$ above. Indeed, we have proved that T is a specialization of $S_{\mathcal{I}}$.
Let us now consider the asserted commutativity result. Since $(S_{\mathcal{I}_1})_{\mathcal{I}_2}$ is $\mathcal{I}_2$-consistent by definition we have

$$\mathcal{I}_2 \quad \Rightarrow \quad wlp((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\mathcal{I}_2) \quad .$$

On the other side we can use the definition of GCS and consistency as well as $(S_{\mathcal{I}_1})_{\mathcal{I}_2} \sqsubseteq S_{\mathcal{I}_1}$ in order to receive

$$\mathcal{I}_1 \quad \Rightarrow \quad wlp(S_{\mathcal{I}_1})(\mathcal{I}_1) \quad \Rightarrow \quad wlp((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\mathcal{I}_1) \quad .$$

In summary, this results in

$$\mathcal{I}_1 \wedge \mathcal{I}_2 \;\Rightarrow\; wlp((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\mathcal{I}_1) \wedge wlp((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\mathcal{I}_2) \;\Leftrightarrow\; wlp((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\mathcal{I}_1 \wedge \mathcal{I}_2) \quad ,$$

so we have proved the consistency of $(S_{\mathcal{I}_1})_{\mathcal{I}_2}$ with respect to $\mathcal{I}_1 \wedge \mathcal{I}_2$. From $S_{\mathcal{I}_1} \sqsubseteq S$ and $(S_{\mathcal{I}_1})_{\mathcal{I}_2} \sqsubseteq S_{\mathcal{I}_1}$ we derive

$$wlp(S)(\varphi) \quad \Rightarrow \quad wlp(S_{\mathcal{I}_1})(\varphi) \quad \Rightarrow \quad wlp((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\varphi),$$

i.e. the specialization $(S_{\mathcal{I}_1})_{\mathcal{I}_2} \sqsubseteq S$. Consequentely, definition 4 yields $(S_{\mathcal{I}_1})_{\mathcal{I}_2} \sqsubseteq S_{\mathcal{I}_1 \wedge \mathcal{I}_2}$ and we obtain

$$\begin{array}{rcl} wlp(\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\varphi) & \Leftrightarrow & \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow wlp(S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\varphi) \\ & \Rightarrow & \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow wlp((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\varphi) \\ & \Leftrightarrow & wlp(\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow (S_{\mathcal{I}_1})_{\mathcal{I}_2})(\varphi) \end{array}$$

for arbitrary φ which means $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow (S_{\mathcal{I}_1})_{\mathcal{I}_2} \sqsubseteq \mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2}$. Thus, it remains to show the reverse specialization.
From $S_{\mathcal{I}_1 \wedge \mathcal{I}_2} \sqsubseteq S$ follows

$$\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2} \quad \sqsubseteq \quad S \quad . \tag{1}$$

In addition, $S_{\mathcal{I}_1 \wedge \mathcal{I}_2}$ is consistent with respect to $\mathcal{I}_1 \wedge \mathcal{I}_2$ of definition, so we have not only $\mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow wlp(S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\mathcal{I}_1)$ but also $\mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow wlp(S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\mathcal{I}_2)$. Next we consider

$$\mathcal{I}_1 \Rightarrow wlp(\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\mathcal{I}_1) \quad \Leftrightarrow \quad \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow wlp(S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\mathcal{I}_1) \tag{2}$$

and

$$\mathcal{I}_2 \Rightarrow wlp(\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\mathcal{I}_2) \quad \Leftrightarrow \quad \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow wlp(S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\mathcal{I}_2) \quad . \tag{3}$$

From equation (2) we obtain the consistency of $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2}$ with respect to $\mathcal{I}_1$ and using equation (1) yields

$$\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2} \quad \sqsubseteq \quad S_{\mathcal{I}_1} \quad . \tag{4}$$

From equation (3) follows the consistency of $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2}$ with respect to $\mathcal{I}_2$ and using equation (4) we conclude

$$\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2} \quad \sqsubseteq \quad (S_{\mathcal{I}_1})_{\mathcal{I}_2} \quad . \tag{5}$$

Finally, we compute

$$\begin{array}{rcl} w(l)p(\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow (S_{\mathcal{I}_1})_{\mathcal{I}_2})(\varphi) & \Leftrightarrow & \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow w(l)p((S_{\mathcal{I}_1})_{\mathcal{I}_2})(\varphi) \\ & \Rightarrow & \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow w(l)p(\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\varphi) \\ & \Leftrightarrow & \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow (\mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow w(l)p(S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\varphi)) \\ & \Leftrightarrow & \mathcal{I}_1 \wedge \mathcal{I}_2 \Rightarrow w(l)p(S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\varphi) \\ & \Leftrightarrow & w(l)p(\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2})(\varphi) \end{array}$$

the specialization $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2} \sqsubseteq \mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow (S_{\mathcal{I}_1})_{\mathcal{I}_2}$, where we just make use of equation (5) in the appearing implication. This completes the proof. □

C Appendix C: Proof of the Upper Bound Theorem

Recall the strategy, to obtain a new specification $S'_{\mathcal{I}}$ from a given complex program specification S and static invariant $\mathcal{I}$ by replacing all basic commands, i.e. *skip*, *fail*, *loop* and in particular assignments, within S by their respective GCSs. The upper bound theorem 1 proposes that this yields an upper bound for $S_{\mathcal{I}}$ with respect to the specialization order $\sqsubseteq$, i.e., $S_{\mathcal{I}} \sqsubseteq S'_{\mathcal{I}}$.

The result is only provable if we assume that S is in $\mathcal{I}$-reduced form. We use structural induction on guarded commands and start with $\rightarrow, \Box, @$ and $\boxtimes$. We will deal with the more difficult cases of sequences and recursion in subsections.

Proposition 7 *Let $S' = P \rightarrow S$ be a specification on Y and $\mathcal{I}$ a static constraint on X with $Y \subseteq X$. If $T \sqsubseteq S'$ is $\mathcal{I}$-consistent, then $T \sqsubseteq P \rightarrow S_{\mathcal{I}}$.*

Proof. First $w(l)p(S)(\varphi) \Rightarrow (P \Rightarrow w(l)p(S)(\varphi))$ establishes $S' \sqsubseteq S$, hence $T \sqsubseteq S$ by assumption and transitivity of $\sqsubseteq$. Moreover, the $\mathcal{I}$-consistency of T gives us even $T \sqsubseteq S_{\mathcal{I}}$. From

$$wp(S')(false) \quad \Leftrightarrow \quad P \Rightarrow wp(S)(false) \quad \Leftrightarrow \quad \neg P \vee wp(S)(false)$$

we receive $\neg P \Rightarrow wp(S')(false)$. As the specialization $T \sqsubseteq S'$ means in particular $wp(S')(false) \Rightarrow wp(T)(false)$, we conclude $\neg P \Rightarrow wp(T)(false)$ or equivalently $wp(T)^*(true) \Rightarrow P$. But then

$$\begin{array}{rcl} w(l)p(\mathcal{P} \rightarrow S_{\mathcal{I}})(\varphi) & \Leftrightarrow & P \Rightarrow w(l)p(S_{\mathcal{I}})(\varphi) \\ & \Rightarrow & P \Rightarrow w(l)p(T)(\varphi) \\ & \Rightarrow & wp(T)^*(true) \Rightarrow w(l)p(T)(\varphi) \\ & \Leftrightarrow & w(l)p(wp(T)^*(true) \rightarrow T)(\varphi) \\ & \Leftrightarrow & w(l)p(T)(\varphi) \quad , \end{array}$$

holds and therefore the desired specialization $T \sqsubseteq P \to S_{\mathcal{I}}$. □

Proposition 8 *Let $S = S_1 \Box S_2$ be a program specification on Y and $\mathcal{I}$ a static invariant on X with $Y \subseteq X$. If $T \sqsubseteq S$ is $\mathcal{I}$-consistent, then $T \sqsubseteq (S_1)_{\mathcal{I}} \Box (S_2)_{\mathcal{I}}$.*

Proof. We start showing the semantic equivalence of T to $T' \Box \mathcal{Q} \to loop$ with $wp(T')(true) \Leftrightarrow true$, $wlp(T')(\varphi) \Leftrightarrow wlp(T)(\varphi)$ for arbitrary φ and $\mathcal{Q} \Leftrightarrow wp(T)^*(false)$. Namely,

$$\begin{array}{rcl} wlp(T'\Box\mathcal{Q} \to loop)(\varphi) & \Leftrightarrow & wlp(T')(\varphi) \wedge (\mathcal{Q} \Rightarrow wlp(loop)(\varphi)) \\ & \Leftrightarrow & wlp(T)(\varphi) \wedge true \\ & \Leftrightarrow & wlp(T)(\varphi) \quad \text{and} \\ wp(T'\Box\mathcal{Q} \to loop)(\varphi) & \Leftrightarrow & wp(T')(\varphi) \wedge (\mathcal{Q} \Rightarrow wp(loop)(\varphi)) \\ & \Leftrightarrow & wlp(T')(\varphi) \wedge wp(T')(true) \wedge \neg\mathcal{Q} \\ & \Leftrightarrow & wlp(T)(\varphi) \wedge \neg wp(T)^*(false) \\ & \Leftrightarrow & wlp(T)(\varphi) \wedge wp(T)(true) \\ & \Leftrightarrow & wp(T)(\varphi). \end{array}$$

From

$$w(l)p(S)(\varphi) \Rightarrow w(l)p(T)(\varphi) \quad \Leftrightarrow \quad w(l)p(T')(\varphi) \wedge w(l)p(\mathcal{Q} \to loop)(\varphi)$$

we obtain $\mathcal{Q} \to loop \sqsubseteq S$ and therefore also

$$\mathcal{Q} \to loop \;=\; (\mathcal{Q}_1 \to loop)\Box(\mathcal{Q}_2 \to loop),$$

with $\mathcal{Q}_i \to loop \sqsubseteq S_i$ for $i = 1, 2$. We show $T' \sqsubseteq (S_1)_{\mathcal{I}} \Box (S_2)_{\mathcal{I}}$ since this implies

$$T \;\sqsubseteq\; \underbrace{(S_1)_{\mathcal{I}}\Box(\mathcal{Q}_1 \to loop)}_{(S_1)'_{\mathcal{I}}} \Box \underbrace{(S_2)_{\mathcal{I}}\Box(\mathcal{Q}_2 \to loop)}_{(S_2)'_{\mathcal{I}}}$$

with $(S_i)'_{\mathcal{I}} \sqsubseteq (S_i)_{\mathcal{I}}$ for $i = 1, 2$. Namely, $\mathcal{Q}_i \to loop \sqsubseteq S_i$, $(S_i)_{\mathcal{I}} \sqsubseteq S_i$ implies $(S_i)'_{\mathcal{I}} \sqsubseteq S_i$ and from the $\mathcal{I}$-consistency of $(S_i)'_{\mathcal{I}}$ follows $(S_i)'_{\mathcal{I}} \sqsubseteq (S_i)_{\mathcal{I}}$.

Without loss in generality we assume that $wp(T)(true) \Leftrightarrow true$ holds. For each state $\vec{a}$ on Y we define $T^{\vec{a}} = T; (\vec{y} = \vec{a} \to skip)$. Then $T^{\vec{a}}$ is a deterministic specialization of T as

$$\begin{array}{rcl} wlp(T^{\vec{a}})^*(\vec{y} = \vec{b}) & \Leftrightarrow & wlp(T)^*(\vec{y} = \vec{a} \wedge \vec{y} = \vec{b}) \\ & \Leftrightarrow & \begin{cases} wlp(T)^*(\vec{y} = \vec{a}) & \text{for } \vec{b} = \vec{a} \\ false & \text{otherwise} \end{cases} \\ & \Rightarrow & wlp(T)^*(\vec{y} = \vec{b}) \quad \text{and} \\ wp(T^{\vec{a}})^*(\vec{y} = \vec{b}) & \Leftrightarrow & wp(T)^*(\vec{y} = \vec{a} \wedge \vec{y} = \vec{b}) \\ & \Leftrightarrow & \begin{cases} wp(T)^*(\vec{y} = \vec{a}) & \text{for } \vec{b} = \vec{a} \\ wp(T)^*(false) & \text{otherwise} \end{cases} \\ & \Rightarrow & wp(T)^*(\vec{y} = \vec{b}) \quad . \end{array}$$

The last implication in the second case follows from the monotonicity of $wlp(T)^*$ applied to $false \Rightarrow \vec{y} = \vec{b}$. Besides, we obtain $w(l)p(T^{\vec{a}})^*(R) \Rightarrow w(l)p(T)^*(R)$ for arbitrary φ. From $\varphi(\vec{y}) \Leftrightarrow \forall\vec{z}.\neg\varphi(\vec{z}) \Rightarrow \vec{y} \neq \vec{z}$ we derive

$$\begin{array}{lcl} wlp(T)(\varphi(\vec{y})) & \Leftrightarrow & wlp(T)(\forall\vec{z}.\neg\varphi(\vec{z}) \Rightarrow \vec{y} \neq \vec{z}) \\ & \Leftrightarrow & \forall\vec{z}.\neg\varphi(\vec{z}) \Rightarrow wlp(T)(\vec{y} \neq \vec{z}) \\ & \Rightarrow & \forall\vec{z}.\neg\varphi(\vec{z}) \Rightarrow wlp(T^{\vec{a}})(\vec{y} \neq \vec{z}) \\ & \Leftrightarrow & wlp(T^{\vec{a}})(\forall\vec{z}.\neg\varphi(\vec{z}) \Rightarrow \vec{y} \neq \vec{z}) \\ & \Leftrightarrow & wlp(T^{\vec{a}})(\varphi(\vec{y})) \quad , \end{array}$$

i.e., the specialization $T^{\vec{a}} \sqsubseteq T$, as the wp-part can be obtain similarily. Here, in case of an empty index set we use $wp(T^{\vec{a}})(true) \Leftrightarrow wp(T)(true)$. The proof that $T^{\vec{a}}$ is deterministic uses

$$wlp(T^{\vec{a}})^*(\vec{y} = \vec{b}) \quad \Leftrightarrow \quad wlp(T)^*(\vec{y} = \vec{b} \wedge \vec{y} = \vec{a})$$

and the distinction into two cases. If $\vec{b} \neq \vec{a}$ holds, then

$$wlp(T)^*(\vec{y} = \vec{b} \wedge \vec{y} = \vec{a}) \Leftrightarrow wlp(T)^*(false) \Leftrightarrow false \Rightarrow wp(T^{\vec{a}})(\vec{y} = \vec{b})$$

and if $\vec{b} = \vec{a}$ is valid, then

$$wp(T^{\vec{a}})(\vec{y} = \vec{a}) \Leftrightarrow wp(T)(\vec{y} = \vec{a} \Rightarrow \vec{y} = \vec{a}) \Leftrightarrow true$$

implies $wp(T^{\vec{a}})(\vec{y} = \vec{b})$. Together $wlp(T^{\vec{a}})^*(\varphi) \Rightarrow wp(T^{\vec{a}})(\varphi)$ for arbitrary φ means that $T^{\vec{a}}$ is deterministic. Using wlp's monotonicity, we conclude $\mathcal{I} \Rightarrow wlp(T)(\mathcal{I}) \Rightarrow wlp(T)(\vec{y} = \vec{a} \Rightarrow \mathcal{I}) \Rightarrow wlp(T^{\vec{a}})(\mathcal{I})$ and therefore that $T^{\vec{a}}$ is also $\mathcal{I}$-consistent. As we have just proven that $T^{\vec{a}}$ is deterministic, it is also semantically equivalent to $T_1^{\vec{a}} \Box T_2^{\vec{a}}$ with $T_i^{\vec{a}} \sqsubseteq S_i$ for $i = 1, 2$. More precisely, we have $T_i^{\vec{a}} = P_i^{\vec{a}} \rightarrow T^{\vec{a}}$ with

$$P_i^{\vec{a}} \Leftrightarrow \{\vec{z}/\vec{y}\}.wlp(\{\vec{y}/\vec{z}\}.T^{\vec{a}})(\vec{z} = \vec{a} \Rightarrow wlp(S_i)^*(\vec{z} = \vec{y})).$$

Using Proposition 3 we have

$$P_1^{\vec{a}} \vee P_2^{\vec{a}} \Leftrightarrow \{\vec{z}/\vec{y}\}.wlp(\{\vec{y}/\vec{z}\}.T^{\vec{a}})(\vec{y} = \vec{a} \Rightarrow wlp(S)^*(\vec{z} = \vec{y})) \Leftrightarrow true \quad ,$$

where $T^{\vec{a}} \sqsubseteq S$ is applied. Moreover, $T^{\vec{a}} = (P_1^{\vec{a}} \vee P_2^{\vec{a}}) \rightarrow T^{\vec{a}} = P_1^{\vec{a}} \rightarrow T^{\vec{a}} \Box P_2^{\vec{a}} \rightarrow T^{\vec{a}}$ holds. Since $T_i^{\vec{a}}$ is $\mathcal{I}$-consistent for $i = 1, 2$, the GCS definition gives us $T_i^{\vec{a}} \sqsubseteq (S_i)_{\mathcal{I}}$ and therefore $T^{\vec{a}} \sqsubseteq (S_1)_{\mathcal{I}} \Box (S_2)_{\mathcal{I}}$.

Finally, the least upper bound of all $T^{\vec{a}}$ with respect to $\sqsubseteq$ must be a specialization of $(S_1)_{\mathcal{I}} \Box (S_2)_{\mathcal{I}}$. But this least upper bound is T and the proof is done. $\Box$

The case of unbounded choice can be proven similarily to the last case.

Proposition 9 *Let $S' = @y \bullet S$ be a specification on Y and $\mathcal{I}$ a static constraint on X with $Y \subseteq X$. If $T \sqsubseteq S'$ is $\mathcal{I}$-consistent, then $T \sqsubseteq @y \bullet S_{\mathcal{I}}$.* $\Box$

Proposition 10 *Let $S = S_1 \boxtimes S_2$ a specification on Y and $\mathcal{I}$ a static constraint on X with $Y \subseteq X$. If $T \sqsubseteq S$ is $\mathcal{I}$ consistent, then*

$$T \sqsubseteq (S_1)_{\mathcal{I}} \Box wp(S_1)(false) \to (S_2)_{\mathcal{I}} \sqsubseteq (S_1)_{\mathcal{I}} \boxtimes (S_2)_{\mathcal{I}}.$$

Futhermore, $T \sqsubseteq (S_1)_{\mathcal{I}} \Box wlp(S_1)(false) \to (S_2)_{\mathcal{I}}$ holds.

Proof. We define $T_1 = wp(S_1)^*(true) \to T$ and $T_2 = wp(S_1)(false) \to T$. Let φ be an arbitrary Y-formula. Then we have

$$\begin{array}{lll} w(l)p(T_1 \Box T_2)(\varphi) & \Leftrightarrow & (wp(S_1)^*(true) \wedge w(l)p(T)^*(\varphi)) \vee \\ & & (wp(S_1)(false) \wedge w(l)p(T)^*(\varphi)) \\ & \Leftrightarrow & (wp(S_1)^*(true) \vee wp(S_1)(false)) \wedge w(l)p(T)^*(\varphi) \\ & \Leftrightarrow & w(l)p(T)^*(\varphi) \quad , \end{array}$$

that is T and $T_1 \Box T_2$ are semantically equivalent. By assumption $T \sqsubseteq S_1 \boxtimes S_2$ holds, and hence

$$w(l)p(T)^*(\varphi) \quad \Rightarrow \quad w(l)p(S_1)^*(\varphi) \vee (wp(S_1)(false) \wedge w(l)p(S_2)^*(\varphi))$$

is valid, too. Besides, we can proof

$$\begin{array}{lll} w(l)p(T_1)^*(\varphi) & \Leftrightarrow & wp(S_1)^*(true) \wedge w(l)p(T)^*(\varphi) \\ & \Rightarrow & (wp(S_1)^*(true) \wedge w(l)p(S_1)^*(\varphi)) \vee \\ & & (\underbrace{wp(S_1)^*(true) \wedge wp(S_1)(false)}_{\Leftrightarrow false} \wedge w(l)p(S_2)^*(\varphi)) \\ & \Rightarrow & w(l)p(S_1)(\varphi) \quad . \end{array}$$

But this means $T_1 \sqsubseteq S_1$. Even more,

$$\begin{array}{lll} w(l)p(T_2)^*(\varphi) & \Leftrightarrow & wp(S_1)(false) \wedge w(l)p(T)^*(\varphi) \\ & \Rightarrow & (wp(S_1)(false) \wedge w(l)p(S_1)^*(\varphi)) \vee \\ & & (wp(S_1)(false) \wedge w(l)p(S_2)^*(\varphi)) \\ & \Rightarrow & (\underbrace{wp(S_1)(false) \wedge wp(S_1)^*(true)}_{\Leftrightarrow false}) \vee \\ & & (wp(S_1)(false) \wedge w(l)p(S_2)^*(\varphi)) \\ & \Leftrightarrow & w(l)p(wp(S_1)(false) \to S_2)^*(\varphi) \end{array}$$

gives us $T_2 \sqsubseteq wp(S_1)(false) \to S_2$. Herein, the second implication is due to $\neg wp(S_1)^*(true) \Rightarrow \neg wp(S_1)^*(\varphi)$ and $wlp(S_1)^*(\varphi) \Rightarrow wp(S_1)^*(\varphi)$.
As T_1 and T_2 are $\mathcal{I}$-consistent, we have $T_1 \sqsubseteq (S_1)_{\mathcal{I}}$. Due to Proposition 7 and $(S_1)_{\mathcal{I}} \sqsubseteq S_1$, we derive $T_2 \sqsubseteq wp(S_1)(false) \to (S_2)_{\mathcal{I}} \sqsubseteq wp((S_1)_{\mathcal{I}})(false) \to (S_2)_{\mathcal{I}}$, i.e., by definition of the predicate transformers

$$\begin{array}{lll} T_1 \Box T_2 & \sqsubseteq & (S_1)_{\mathcal{I}} \Box wp(S_1)(false) \to (S_2)_{\mathcal{I}} \\ & \sqsubseteq & (S_1)_{\mathcal{I}} \Box wp((S_1)_{\mathcal{I}})(false) \to (S_2)_{\mathcal{I}} \; = \; (S_1)_{\mathcal{I}} \boxtimes (S_2)_{\mathcal{I}}. \end{array}$$

This gives the first statement, the second one becomes obvious when we look at $wp(S_1)(false) \Rightarrow wlp(S_1)(false)$. □

C.1 The Case for Sequences

We come now to the case of sequences. Herein, the definition of δ-$\mathcal{I}$-reducedness will become more apparent. But first, we will show the following lemma.

Lemma 9 *Let $S = S_1; S_2$ be a program specification on Y with S_i on $Y_i \subseteq Y$ for $i = 1, 2$. Let $\mathcal{I}$ be a static invariant on $X = \{x_1, \dots, x_s\}$ with $Y \subseteq X$. Besides, $X - Y_1 - Y_2 = \{y_1, \dots, y_m\}$, $X - Y_1 = \{y_1, \dots, y_m, y_{m+1}, \dots, y_n\}$, $X - Y_2 = \{y_1, \dots, y_m, x_{l+1}, \dots, x_k\}$, $Y_1 = \{x_1, \dots, x_l, x_{l+1}, \dots, x_k\}$ and $\{x'_1, \dots, x'_k\}$ a disjoint copy of Y_1 with $Y'_1 \cap Y = \emptyset$. If S is δ-$\mathcal{I}$-reduced and S_1 deterministic, then*

1. *for all states $\vec{a}$ and $\vec{b}$ with $\models_{\vec{a}} \neg\mathcal{I}$, $\models_{\vec{b}} \neg\mathcal{I}$ and $\models_{\vec{a}} wlp(S)^*(\exists y_1, \dots, y_m. \vec{x} = \vec{b})$, for which*
$$\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall y_1, \dots, y_n.wlp(S_2)^*(\exists y_1, \dots, y_m, x_{l+1}, \dots, x_k.\vec{x} = \vec{b}) \Rightarrow \mathcal{I})$$
is a δ-constraint for S_1, $\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.\forall y_1, \dots, y_n.\mathcal{I}$ is a δ-constraint for S.

2. *for all states $\vec{a}$ and $\vec{b}$ with $\models_{\vec{a}} \mathcal{I}$, $\models_{\vec{b}} \mathcal{I}$ and $\models_{\vec{a}} wlp(S)^*(\exists y_1, \dots, y_m. \vec{x} = \vec{b})$, for which*
$$\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall y_1, \dots, y_n.wlp(S_2)^*(\exists y_1, \dots, y_m, x_{l+1}, \dots, x_k.\vec{x} = \vec{b}) \Rightarrow \neg\mathcal{I})$$
is a δ-constraint for S_1, $\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.\forall y_1, \dots, y_n.\neg\mathcal{I}$ is a δ-constraint for S.

Proof. We will show (i) only. The proof for (ii) is completely analogously. Let $\vec{a}$ and $\vec{b}$ be states with $\models_{\vec{a}} \neg\mathcal{I}$, $\models_{\vec{b}} \neg\mathcal{I}$ und $\models_{\vec{a}} wlp(S)^*(\exists y_1, \dots, y_m. \vec{x} = \vec{b})$ and

$$\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall y_1, \dots, y_n.wlp(S_2)^*(\exists y_1, \dots, y_m, x_{l+1}, \dots, x_k.\vec{x} = \vec{b}) \Rightarrow \mathcal{I}) \quad (*)$$

a δ-constraint for S_1. Then

$$\begin{aligned} \models_{\vec{a}} wlp(S)^*(\exists y_1, \dots, y_m.\vec{x} = \vec{b}) &\Leftrightarrow \models_{\vec{a}} wlp(S_1)^*(wlp(S_2)^*(\exists y_1, \dots, y_m.\vec{x} = \vec{b})) \\ &\Rightarrow \models_{\vec{a}} wp(S_1)(wlp(S_2)^*(\exists y_1, \dots, y_m.\vec{x} = \vec{b})) \\ &\Rightarrow \models_{\vec{a}} wlp(S_1)(wlp(S_2)^*(\exists y_1, \dots, y_m.\vec{x} = \vec{b})) \end{aligned}$$

holds, using the definition of $wlp(S)$, S_1 being deterministic and the pairing condition. Moreover, we conclude

$$\models \{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S_1)(\vec{x} = \vec{a} \Rightarrow wlp(\{\vec{x}/\vec{x}'\}.S_2)^*(\exists y_1, \dots, y_m.\{\vec{x}/\vec{x}'\}.\vec{x} = \vec{b})) \quad (**)$$

by definition of $wlp(S_1)$. By definition of a δ-constraint, $(*)$ implies

$$\models \{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S_1)(\vec{x} = \vec{a} \Rightarrow \\ \{\vec{x}/\vec{x}'\}.\forall y_1, \dots, y_n.wlp(S_2)^*(\exists y_1, \dots, y_m, x_{l+1}, \dots, x_k.\vec{x} = \vec{b}) \Rightarrow \mathcal{I}))$$

and together with $(**)$ further

$$\models \{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S_1)(\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.\forall y_1, \dots, y_n.\mathcal{I}) \quad .$$

Hence, $\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.\forall y_1, \dots, y_n.\mathcal{I}$ is a δ-constraint for S_1. As S is δ-$\mathcal{I}$-reduced by assumption, $\vec{x} = \vec{a} \Rightarrow \{\vec{x}/\vec{x}'\}.\forall y_1, \dots, y_n.\mathcal{I}$ is also a δ-constraint for S. □

Proposition 11 *Let $S = S_1; S_2$ be an $\mathcal{I}$-reduced specification on Y with $\mathcal{I}$ being a static constraint on X with $Y \subseteq X$. If $T \sqsubseteq S$ is $\mathcal{I}$-consistent, then $T \sqsubseteq (S_1)_\mathcal{I}; (S_2)_\mathcal{I}$.*

Proof. Without loss in generality we assume that $wp(T)(true) \Leftrightarrow true$ holds. Then it suffices to show $wlp(S_\mathcal{I})^*(\vec{x} = \vec{a}) \Rightarrow wlp((S_1)_\mathcal{I}; (S_2)_\mathcal{I})^*(\vec{x} = \vec{a})$ for all state characterising formulae $\vec{x} = \vec{a}$. Namely,

$$\begin{array}{rcl} wlp((S_1)_\mathcal{I})(wlp((S_2)_\mathcal{I})(\varphi(\vec{x}))) & \Leftrightarrow & wlp((S_1)_\mathcal{I})(wlp((S_2)_\mathcal{I})(\forall \vec{z}.\neg\varphi(\vec{z}) \Rightarrow \vec{x} \neq \vec{z})) \\ & \Leftrightarrow & wlp((S_1)_\mathcal{I})(\forall \vec{z}.\neg\varphi(\vec{z}) \Rightarrow wlp((S_2)_\mathcal{I})(\vec{x} \neq \vec{z})) \\ & \Leftrightarrow & \forall \vec{z}.\neg\varphi(\vec{z}) \Rightarrow wlp((S_1)_\mathcal{I})(wlp((S_2)_\mathcal{I})(\vec{x} \neq \vec{z})) \\ & \Rightarrow & \forall \vec{z}.\neg\varphi(\vec{z}) \Rightarrow wlp(S_\mathcal{I})(\vec{x} \neq \vec{z}) \\ & \Leftrightarrow & wlp(S_\mathcal{I})(\forall \vec{z}.\neg\varphi(\vec{z}) \Rightarrow \vec{x} \neq \vec{z}) \\ & \Leftrightarrow & wlp(S_\mathcal{I})(\varphi(\vec{x})) \end{array}$$

holds for all X-formulae φ.

As S_1 is the least upper bound of its deterministic branches with respect to $\sqsubseteq$, we can further assume without loss in generality that S_1 is deterministic. Therefore, we are able to use the stronger properties from Lemma 9.

First, we compute both sides of of the implication above using the GCS normal form from Proposition 4. We obtain

$$wlp(S_\mathcal{I})^*(\vec{x} = \vec{a}) \Leftrightarrow \boxed{\begin{array}{c} (\mathcal{I} \wedge \exists \vec{\xi}.wlp(S)^*(\{\vec{y}/\vec{\xi}\}.\mathcal{I} \wedge \{\vec{y}/\vec{\xi}\}.\vec{x} = \vec{a})) \vee \\ (\neg\mathcal{I} \wedge \exists \vec{\xi}.wlp(S)^*(\{\vec{y}/\vec{\xi}\}.\vec{x} = \vec{a})) \end{array}} \qquad (6)$$

as well as

$$\begin{aligned}
wlp((S_1)_{\mathcal{I}};(S_2)_{\mathcal{I}})^*(\vec{x}=\vec{a}) &\Leftrightarrow (\mathcal{I}\wedge wlp(S_1)^*(\exists y_1,\dots,y_n.(\mathcal{I}\wedge \\
&\qquad wlp((S_2)_{\mathcal{I}})^*(\vec{x}=\vec{a})))) \vee (\neg\mathcal{I}\wedge wlp(S_1)^*(\\
&\qquad \exists y_1,\dots,y_n.wlp((S_2)_{\mathcal{I}})^*(\vec{x}=\vec{a}))) \\
&\Leftrightarrow (\mathcal{I}\wedge wlp(S_1)^*(\exists y_1,\dots,y_n.(\mathcal{I}\wedge wlp(S_2)^*(\\
&\qquad \exists y_1,\dots,y_m,x_{l+1},\dots,x_k(\mathcal{I}\wedge\vec{x}=\vec{a}))))) \vee \\
&\quad (\neg\mathcal{I}\wedge wlp(S_1)^*(\exists y_1,\dots,y_n.(\mathcal{I}\wedge wlp(S_2)^*(\\
&\qquad \exists y_1,\dots,y_m,x_{l+1},\dots,x_k(\mathcal{I}\wedge\vec{x}=\vec{a}))))) \vee \\
&\quad (\neg\mathcal{I}\wedge wlp(S_1)^*(\exists y_1,\dots,y_n.(\neg\mathcal{I}\wedge \\
&\quad wlp(S_2)^*(\exists y_1,\dots,y_m,x_{l+1},\dots,x_k.\vec{x}=\vec{a}))))
\end{aligned}$$

and this is equivalent to

$$\boxed{\begin{array}{c}
\exists\xi_1,\dots,\xi_n.\exists\xi'_1,\dots,\xi'_m,\xi'_{l+1},\dots,\xi'_k.(wlp(S_1)^*(\{\vec{y}/\vec{\xi}\}.\mathcal{I}\wedge \\
\{\vec{y}/\vec{\xi}\}.wlp(S_2)^*(\{\vec{y}/\vec{\xi'}\}.(\mathcal{I}\wedge\vec{x}=\vec{a}))))\vee \\
\exists\xi_1,\dots,\xi_n.\exists\xi'_1,\dots,\xi'_m,\xi'_{l+1},\dots,\xi'_k.\neg\mathcal{I}\wedge \\
(wlp(S_1)^*(\neg\{\vec{y}/\vec{\xi}\}.\mathcal{I}\wedge\{\vec{y}/\vec{\xi}\}.wlp(S_2)^*(\{\vec{y}/\vec{\xi'}\}.\vec{x}=\vec{a})))
\end{array}} \qquad (7)$$

Case 1. We assume $\vec{x}=\vec{a}\Rightarrow\neg\mathcal{I}$. Then $wlp(S_{\mathcal{I}})^*(\vec{x}=\vec{a})\Rightarrow wlp(S_{\mathcal{I}})^*(\neg\mathcal{I})\Rightarrow\neg\mathcal{I}$ follows as $S_{\mathcal{I}}$ is $\mathcal{I}$-consistent. Since we also $wlp(S_{\mathcal{I}})^*(\vec{x}=\vec{a})$ assume, we look at the second line of formula (6). We show, that we can derive the second subformula of (7). Assuming consistency, we are allowed to neglect $\neg\mathcal{I}$, i.e., we need to derive

$$\begin{aligned}
&\exists\xi_1,\dots,\xi_n.\exists\xi'_1,\dots,\xi'_m,\xi'_{l+1},\dots,\xi'_k.(\neg\mathcal{I}\wedge \\
&\quad (wlp(S_1)^*(\neg\{\vec{y}/\vec{\xi}\}.\mathcal{I}\wedge\{\vec{y}/\vec{\xi}\}.wlp(S_2)^*(\{\vec{y}/\vec{\xi'}\}.\vec{x}=\vec{a})))) \qquad (8)
\end{aligned}$$

Suppose, (8) does not hold. Then, there is a state $\vec{b}$ with

$$\begin{aligned}
&\models_{\vec{b}} wlp(S_1)(\forall\xi_1,\dots,\xi_n.\{\vec{y}/\vec{\xi}\}.(wlp(S_2)^* \\
&\qquad (\exists\xi'_1,\dots,\xi'_m,\xi'_{l+1},\dots,\xi'_k.\{\vec{y}/\vec{\xi'}\}.\vec{x}=\vec{a})\Rightarrow\mathcal{I}))). \qquad (9)
\end{aligned}$$

We compute that (9) is equivalent to

$$\begin{gathered}
\models_{\vec{b}}\{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S_1)(\{\vec{x}/\vec{x}'\}. \\
\underbrace{\forall\xi_1,\dots,\xi_n.\{\vec{y}/\vec{\xi}\}.(wlp(S_2)^*(\exists\xi'_1,\dots,\xi'_m,\xi'_{l+1},\dots,\xi'_k.\{\vec{y}/\vec{\xi'}\}.\vec{x}=\vec{a})\Rightarrow\mathcal{I}))}_{R}
\end{gathered}$$

and therefore to

$$\models_{\vec{b}}\{\vec{x}'/\vec{x}\}.(\forall\vec{x}''.wlp(\{\vec{x}/\vec{x}'\}.S_1)^*(\vec{x}'=\vec{x}'')\Rightarrow\{\vec{x}'/\vec{x}''\}.\{\vec{x}/\vec{x}'\}.R)$$

applying Lemma 1 to $wlp(\{\vec{x}/\vec{x}'\}.S_1)(\{\vec{x}/\vec{x}'\}.R)$. From this, we derive the equivalence to

$$\begin{aligned}&\vec{x} = \vec{b} \Rightarrow \{\vec{x}'/\vec{x}\}.(\forall \vec{x}''.wlp(\{\vec{x}/\vec{x}'\}.S_1)^*(\vec{x}' = \vec{x}'') \Rightarrow \{\vec{x}'/\vec{x}''\}.\{\vec{x}/\vec{x}'\}.R) \Leftrightarrow\\ &\{\vec{x}'/\vec{x}\}.(\forall \vec{x}''.wlp(\{\vec{x}/\vec{x}'\}.S_1)^*(\vec{x}' = \vec{x}'') \Rightarrow (\vec{x} = \vec{b} \Rightarrow \{\vec{x}'/\vec{x}''\}.\{\vec{x}/\vec{x}'\}.R)) \Leftrightarrow\\ &\{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S_1)(\vec{x} = \vec{b} \Rightarrow \{\vec{x}/\vec{x}'\}.R).\end{aligned}$$

But then

$$\begin{aligned}&\vec{x} = \vec{b} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall \xi_1, \ldots, \xi_n.\\ &\{\vec{y}/\vec{\xi}\}.(wlp(S_2)^*(\exists \xi_1', \ldots, \xi_m', \xi_{l+1}', \ldots, \xi_k'.\{\vec{y}/\vec{\xi}'\}.\vec{x} = \vec{a}) \Rightarrow \mathcal{I}))\end{aligned} \qquad (10)$$

is a δ-constraint for S_1. As not only $\models_{\vec{b}} \neg\mathcal{I}$, but also $\models_{\vec{a}} \neg\mathcal{I}$ is valid, Lemma 9 (i) implies that

$$\vec{x} = \vec{b} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall \xi_1, \ldots, \xi_n.\{\vec{y}/\vec{\xi}\}.\mathcal{I}) \qquad (11)$$

is a δ-constraint for S. We conclude

$$\{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S)(\vec{x} = \vec{b} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall \xi_1, \ldots, \xi_n.\{\vec{y}/\vec{\xi}\}.\mathcal{I}))$$

and this is equivalent to

$$\begin{aligned}&\models_{\vec{b}} \{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S)(\{\vec{x}/\vec{x}'\}.(\forall \xi_1, \ldots, \xi_n.\{\vec{y}/\vec{\xi}\}.\mathcal{I})) \Leftrightarrow\\ &\models_{\vec{b}} wlp(S)(\forall y_1, \ldots, y_n.\mathcal{I})\end{aligned} \qquad (12)$$

following a similar computation as above. On the other hand, we apply monotonicity on the assumption $\vec{x} = \vec{a} \Rightarrow \neg\mathcal{I}$ and use $S_{\mathcal{I}} \sqsubseteq S$ to compute

$$\begin{aligned}wlp(S_{\mathcal{I}})^*(\vec{x} = \vec{a}) \quad &\Rightarrow \quad wlp(S_{\mathcal{I}})^*(\neg\mathcal{I})\\ &\Rightarrow \quad wlp(S)^*(\neg\mathcal{I})\\ &\Rightarrow \quad wlp(S)^*(\exists y_1, \ldots, y_n.\neg\mathcal{I}) \quad .\end{aligned}$$

But this is a contradiction since

$$wlp(S)^*(\exists y_1, \ldots, y_n.\neg\mathcal{I}) \quad \Leftrightarrow \quad \neg wlp(S)(\forall y_1, \ldots, y_n.\mathcal{I})$$

holds.

Case 2. Now we assume $\vec{x} = \vec{a} \Rightarrow \mathcal{I}$ and $\models_{\vec{b}} wlp(S_{\mathcal{I}})^*(\vec{x} = \vec{a})$. Following (6) we distinguish further.

Case 2.1. We suppose $\models_{\vec{b}} \neg\mathcal{I} \wedge \exists\vec{\xi}.wlp(S)^*(\{\vec{y}/\vec{\xi}\}.\vec{x} = \vec{a})$. For state $\vec{b}$ we have

$$\begin{aligned}&\exists\vec{\xi}.wlp(S_1)^*(wlp(S_2)^*(\{\vec{y}/\vec{\xi}\}.\vec{x} = \vec{a})) \Leftrightarrow\\ &\exists\vec{\xi}.wlp(S_1)^*((\mathcal{I} \vee \neg\mathcal{I}) \wedge wlp(S_2)^*(\{\vec{y}/\vec{\xi}\}.\vec{x} = \vec{a})) \Leftrightarrow\\ &\exists\vec{\xi}.wlp(S_1)^*(\mathcal{I} \wedge wlp(S_2)^*(\{\vec{y}/\vec{\xi}\}.(\vec{x} = \vec{a} \wedge \mathcal{I}))) \vee\\ &\exists\vec{\xi}.wlp(S_1)^*(\neg\mathcal{I} \wedge wlp(S_2)^*(\{\vec{y}/\vec{\xi}\}.\vec{x} = \vec{a}))\end{aligned}$$

and therefore (7). This gives the proof of case 2.1.
Case 2.2. We suppose $\models_{\vec{b}} \mathcal{I} \wedge \exists \vec{\xi}.wlp(S)^*(\{\vec{y}/\vec{\xi}\}.(\mathcal{I} \wedge \vec{x} = \vec{a}))$ and show that

$$\models_{\vec{b}} \exists \xi_1, \dots, \xi_n . \exists \xi'_1, \dots, \xi'_m, \xi'_{l+1}, \dots, \xi'_k .(wlp(S_1)^*(\{\vec{y}/\vec{\xi}\}.\mathcal{I} \wedge \{\vec{y}/\vec{\xi}\}.wlp(S_2)^*(\{\vec{y}/\vec{\xi}'\}.(\mathcal{I} \wedge \vec{x} = \vec{a})))) \quad (13)$$

follows. This implies the first subformula in (7). According to case 1, we assume that (13) does not hold. Similar to the computations above, we conclude that

$$\vec{x} = \vec{b} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall \xi_1, \dots, \xi_n . \{\vec{y}/\vec{\xi}\}.(wlp(S_2)^*(\exists \xi'_1, \dots, \xi'_m, \xi'_{l+1}, \dots, \xi'_k . \{\vec{y}/\xi'\}.\vec{x} = \vec{a}) \Rightarrow \neg \mathcal{I}))$$

is a δ-constraint for S_1. Using Lemma 9 (ii) as well as $\models_{\vec{b}} \mathcal{I}$ and $\models_{\vec{a}} \mathcal{I}$, we can conclude that

$$\vec{x} = \vec{b} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall \xi_1, \dots, \xi_n . \{\vec{y}/\vec{\xi}\}.\neg \mathcal{I})$$

is a δ-constraint for S. We derive

$$\{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S)(\vec{x} = \vec{b} \Rightarrow \{\vec{x}/\vec{x}'\}.(\forall \xi_1, \dots, \xi_n . \{\vec{y}/\vec{\xi}\}.\neg \mathcal{I}))$$

and further

$$\models_{\vec{b}} \{\vec{x}'/\vec{x}\}.wlp(\{\vec{x}/\vec{x}'\}.S)(\{\vec{x}/\vec{x}'\}.(\forall \xi_1, \dots, \xi_n . \{\vec{y}/\vec{\xi}\}.\neg \mathcal{I})) \quad ,$$

i.e., equivalence to

$$\models_{\vec{b}} wlp(S)(\forall y_1, \dots, y_n . \neg \mathcal{I}). \quad (14)$$

Due to our assumptions and $\vec{x} = \vec{a} \Rightarrow \mathcal{I}$ we can also conclude that

$$\begin{array}{rcl} wlp(S_{\mathcal{I}})^*(\vec{x} = \vec{a}) & \Rightarrow & wlp(S_{\mathcal{I}})^*(\mathcal{I}) \\ & \Rightarrow & wlp(S)^*(\mathcal{I}) \\ & \Rightarrow & wlp(S)^*(\exists y_1, \dots, y_n . \mathcal{I}) \\ & \Leftrightarrow & \neg wlp(S)(\forall y_1, \dots, y_n . \neg \mathcal{I}) \end{array}$$

holds, a contradiction to (14). This gives the proof for case 2.2. $\square$

C.2 The Recursive Case

In this appendix we prove the upper bound theorem for recursive operations restricted to simple WHILE-loops in the form of $f(S) = \mathcal{P} \rightarrow T; S \square \neg \mathcal{P} \rightarrow skip$ for which we know the existence of least fixpoints according to subsection 3.2. For this we need some additional lemmata.

For recursive guarded commands the monotonicity of all operation constructors with respect to the Nelson-order $\preceq$ is fundamental [4]. Unfortunately, a similiar result does not hold for the specialization order $\sqsubseteq$. More precisely, the result is false for the $\boxtimes$-constructor in its first component.

Lemma 10 *Let $f(S)$ be a guarded-command expression with the program variable S in which restricted choice $\boxtimes$ does not occur. Then f is monotonic with respect to the specialization order $\sqsubseteq$.*

Proof. The proof is done by structural induction. For each constructor it is completely analogous to the corresponding proof for the Nelson-order in [4]. We omit the details. $\square$

In [6, Proposition 20, p.120] we have seen that $S'_{\mathcal{I}}$ may contain the choice-constructor instead of restricted choice, provided we include some guard. Replacing within a recursive operation some $S_1 \boxtimes S_2$ by $(S_1)_{\mathcal{I}} \boxtimes (S_2)_{\mathcal{I}}$ would destroy the required result.

The next lemma follows from taking together the cases in the upper bound theorem for preconditionings $\rightarrow$, choices $\square$, unbounded choices @ and restricted choices $\boxtimes$.

Lemma 11 *Let T be a consistent specialization of some $\mathcal{I}$-reduced $f(S')$ with respect to $\mathcal{I}$, where $f(S)$ is an expression built from the constructors of guarded commands. Construct $f_{\mathcal{I}}(S)$ from $f(S)$ as follows:*

(i) Each restricted choice $S_1 \boxtimes S_2$ occuring within $f(S)$ will be replaced by $S_1 \square wlp(S_1)(false) \rightarrow S_2$.

(ii) Then each basic operation, i.e. skip and assignments will be replaced by their GCSs with respect to $\mathcal{I}$.

Then we have $T \sqsubseteq f_{\mathcal{I}}(S'_{\mathcal{I}})$. $\square$

We must now face the main difficulty to bring together two different partial orders, namely the specialization order $\sqsubseteq$ which is fundamental for GCSs and the Nelson-order $\preceq$ required for recursion.

In order to accomplish this we will need to make use of another limit operator $\lim_{i \in \mathbb{N}} f^i(loop)_{\mathcal{I}}$. Semantics is completely analogously assigned as for the case of $\lim_{i \in \mathbb{N}} f^i(loop)$. Therefore, we receive a corresponding result to Lemma 2 which can be obtained by using Proposition 4. It then is straightforward to verify counterparts for Lemma 3 and Lemma 4, finally.

Lemma 12 *Let $\mathcal{I}$ be a static constraint and $f(T) = \mathcal{P} \rightarrow S; T \square \neg\mathcal{P} \rightarrow skip$ such that T does not occur within S. Then for each $j \in \mathbb{N}$, there exist predicate transformers $\tau_l^{\mathcal{I}}(j)$ and $\tau^{\mathcal{I}}(j)$ on arithmetic predicates such that the following properties are satisfied:*

(i) for each arithmetic predicate $\varphi(\vec{x})$, the results of applying these predicate transformers are arithmetic predicates in i and x, say

$$\chi_j^{1,\mathcal{I}}(i, \vec{x}) = \tau_l^{\mathcal{I}}(j)(\varphi(\vec{x})) \qquad \text{and} \qquad \chi_j^{2,\mathcal{I}}(i, \vec{x}) = \tau^{\mathcal{I}}(j)(\varphi(\vec{x}))$$

(ii) for $j = h(\varphi)$ we obtain

$$\begin{array}{ll} \forall \vec{x}.\forall i. \left(\chi_j^{1,\mathcal{I}}(i,\vec{x}) \Leftrightarrow wlp(f^i(loop))(\varphi(\vec{x}))\right) & \textit{and} \\ \forall \vec{x}.\forall i. \left(\chi_j^{2,\mathcal{I}}(i,\vec{x}) \Leftrightarrow wp(f^i(loop))(\varphi(\vec{x}))\right) & \end{array}$$

with $\vec{x} = x_{i_1}, \ldots, x_{i_k}$.

Proof. We follow closely the proof Lemma of 2 where we obtained a primitve recursive function $\bar{g}$ such that

$$Q_1'(\bar{g}(k), j, \vec{x}) \quad = \quad wlp(f^k(loop))(\varphi(\vec{x}))$$

is satisfied. Herein, $\bar{g}(k)$ gives us the Gödel number of $f^k(loop)$. Using the normal form for GCSs from Proposition 4, we can easily derive a further primitive recursive function s such that the composition of s with $\bar{g}$ yields the Gödel number $(s \circ \bar{g})(k)$ for $f^k(loop)_\mathcal{I}$. Notice, that $loop_\mathcal{I} = loop$ holds. In particular, we obtain

$$q_1^\mathcal{I}((s \circ \bar{g})(k), j, \vec{x}) \quad = \quad wlp(f^k(loop)_\mathcal{I})(\varphi(\vec{x})).$$

Then, we define predicates $Q_1^\mathcal{I}(k,j,\vec{x}) = q_1^\mathcal{I}((s \circ \bar{g})(k), j, \vec{x})$ and an extension of $\bar{Q}^\mathcal{I}(h(\psi), h(\varphi), \vec{x}) \Leftrightarrow ((\mathcal{P} \Rightarrow wlp(S)(\psi)) \wedge (\neg\mathcal{P} \Rightarrow \varphi))$ with $\psi, \varphi \in \mathbb{F}$. We conclude

$$Q_1^\mathcal{I}(k,j,\vec{x}) \quad = \quad q_1^\mathcal{I}((s \circ \bar{g})(k), j, \vec{x}) \quad = \quad \bar{Q}^\mathcal{I}(h(\psi), h(\varphi), \vec{x}) \quad ,$$

where $h(\varphi) = j$ and $\psi(\vec{x}) = wlp(f^{k-1}(loop)_\mathcal{I})(\varphi(\vec{x})) = Q_1^\mathcal{I}(k-1, j, \vec{x})$. In summary, we receive

$$\begin{array}{rcl} Q_1^\mathcal{I}(0,j,\vec{x}) & = & true \quad \text{and} \\ Q_1^\mathcal{I}(k+1,j,\vec{x}) & = & \bar{Q}^\mathcal{I}(h(Q_1^\mathcal{I}(k,j,\vec{x})), j, \vec{x}). \end{array}$$

Now we take $\tau_l^\mathcal{I}(j)(\varphi(\vec{x})) = \chi_j^{1,\mathcal{I}}(k,\vec{x}) = Q_1^\mathcal{I}(k,j,\vec{x})$ and conlcude as in Lemma 2. □

We now define *limit operators* $\lim_{i\in\mathbb{N}} f^i(loop)_\mathcal{I}$ with help of the predicate transformers $\tau_l^\mathcal{I}(j)$ and $\tau^\mathcal{I}(j)$:

$$\begin{array}{llll} wlp\left(\lim\limits_{i\in\mathbb{N}} f^i(loop)_\mathcal{I}\right)(\varphi(\vec{x})) & \Leftrightarrow & \forall i.\chi_{h(\varphi)}^{1,\mathcal{I}}(i,\vec{x}) & \text{and} \\ wp\left(\lim\limits_{i\in\mathbb{N}} f^i(loop)_\mathcal{I}\right)(\varphi(\vec{x})) & \Leftrightarrow & \exists i.\chi_{h(\varphi)}^{2,\mathcal{I}}(i,\vec{x}) & \end{array}$$

for $\chi_{h(\varphi)}^{1,\mathcal{I}}(i,\vec{x}) = \tau_l^\mathcal{I}(h(\varphi))(\varphi(\vec{x}))$ and $\chi_{h(\varphi)}^{2,\mathcal{I}}(i,\vec{x}) = \tau^\mathcal{I}(h(\varphi))(\varphi(\vec{x}))$.

Lemma 13 *The definition of limits* $\lim_{i\in\mathbb{N}} f^i(loop)_\mathcal{I}$ *is sound.*

Proof. The proof follows exactly the one from Lemma 3. We just need to mention that $\{f^i(loop)_\mathcal{I} \mid i \in \mathbb{N}\}$ is a chain with respect to the Nelson-order $\preceq$. As $loop_\mathcal{I} =$

loop is the $\preceq$-minimum, we have $loop_{\mathcal{I}} \preceq f(loop)_{\mathcal{I}}$. Since $\{f^i(loop) \mid i \in \mathbb{N}\}$ is a $\preceq$-chain and therefore $f^i(loop) \preceq f^{i+1}(loop)$ holds, we can finally derive $f^i(loop)_{\mathcal{I}} \preceq f^{i+1}(loop)_{\mathcal{I}}$ (see also Lemma 15). □

The following lemma gives us the corresponding result to Lemma 4. The proof is again completely analogous.

Lemma 14 *The chain* $\{f^i(loop)_{\mathcal{I}} \mid i \in \mathbb{N}\}$ *has a least upper bound, namely* $\lim_{i\in\mathbb{N}} f^i(loop)_{\mathcal{I}}$. □

We are now prepared to bring specialization- and Nelson-order together.

Lemma 15 *Let T und S be Y-operations. Furthermore, let $\mathcal{I}$ be an invariant on X for $Y \subseteq X$. Then we have:*

(i) If $T \preceq S$ holds, then $T_{\mathcal{I}} \preceq S_{\mathcal{I}}$ follows.

(ii) $\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}} \sqsubseteq \lim_{i\in\mathbb{N}}(f^i(loop))_{\mathcal{I}}$.

Proof. (i) Here we use the normal form of a GCS given in Proposition 4. The first result follows immediately, because all constructors are monotonic in the Nelson-order $\preceq$.

(ii) First, $\lim_{i\in\mathbb{N}} f^i(loop)$ is the least upper bound of $\{f^i(loop) \mid i \in \mathbb{N}\}$ with respect to the Nelson-order according to Lemma 4, i.e. especially $f^i(loop) \preceq \lim_{i\in\mathbb{N}} f^i(loop)$ holds for arbitrary $i \in \mathbb{N}$. From this and (i) we get $f^i(loop)_{\mathcal{I}} \preceq \left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}$, i.e. $\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}$ is an upper bound for $\{f^i(loop)_{\mathcal{I}} \mid i \in \mathbb{N}\}$. Using Lemma 14, $\lim_{i\in\mathbb{N}} f^i(loop)_{\mathcal{I}}$ is the least upper bound of the chain $\{f^i(loop)_{\mathcal{I}} \mid i \in \mathbb{N}\}$ which means that $\lim_{i\in\mathbb{N}} f^i(loop)_{\mathcal{I}} \preceq \left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}$ must hold. Therefore, we receive

$$wp\left(\lim_{i\in\mathbb{N}} f^i(loop)_{\mathcal{I}}\right)(\varphi) \quad \Rightarrow \quad wp\left(\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}\right)(\varphi)$$

according to the definition of the Nelson-order.

Once again we make use of Proposition 4 in order to compute

$$\begin{aligned}
& wlp\left(\lim_{i\in\mathbb{N}} f^i(loop)_{\mathcal{I}}\right)(\varphi) \Leftrightarrow \\
& \forall i.i \in \mathbb{N} \Rightarrow wlp\left(f^i(loop)_{\mathcal{I}}\right)(\varphi) \Leftrightarrow \\
& \forall i.i \in \mathbb{N} \Rightarrow wlp((\mathcal{I} \rightarrow f^i(loop); @\vec{z}' \bullet \vec{z} := \vec{z}'; \mathcal{I} \rightarrow skip)\boxtimes \\
& (\neg\mathcal{I} \rightarrow f^i(loop); @\vec{z}' \bullet \vec{z} := \vec{z}'))(\varphi) \Leftrightarrow \\
& \forall i.i \in \mathbb{N} \Rightarrow ((\mathcal{I} \Rightarrow wlp(f^i(loop))(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\mathcal{I} \Rightarrow \varphi))\wedge \\
& (\neg\mathcal{I} \Rightarrow wlp(f^i(loop))(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\varphi))) \Leftrightarrow \\
& (\mathcal{I} \Rightarrow \forall i.i \in \mathbb{N} \Rightarrow wlp(f^i(loop))(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\mathcal{I} \Rightarrow \varphi))\wedge \\
& (\neg\mathcal{I} \Rightarrow \forall i.i \in \mathbb{N} \Rightarrow wlp(f^i(loop))(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\varphi)) \Leftrightarrow \\
& (\mathcal{I} \Rightarrow wlp\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\mathcal{I} \Rightarrow \varphi))\wedge \\
& (\neg\mathcal{I} \Rightarrow wlp\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)(\forall \vec{z}'.\{\vec{z}/\vec{z}'\}.\varphi)) \Leftrightarrow \\
& wlp((\mathcal{I} \rightarrow \lim_{i\in\mathbb{N}} f^i(loop); @\vec{z}' \bullet \vec{z} := \vec{z}'; \mathcal{I} \rightarrow skip)\boxtimes \\
& (\neg\mathcal{I} \rightarrow \lim_{i\in\mathbb{N}} f^i(loop); @\vec{z}' \bullet \vec{z} := \vec{z}'))(\varphi) \Leftrightarrow \\
& wlp\left(\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}\right)(\varphi) \quad ,
\end{aligned}$$

i.e.

$$wlp\left(\lim_{i\in\mathbb{N}} f^i(loop)_{\mathcal{I}}\right)(\varphi) \Rightarrow wlp\left(\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}\right)(\varphi) \quad ,$$

supplies the asserted specialization. □

We are now able to give the main proof.

Proposition 12 Let $S' = \mu T_j.f(T_j)$ with $f(T_j) = \mathcal{P} \rightarrow T; T_j \Box \neg\mathcal{P} \rightarrow skip$ be an $\mathcal{I}$-reduced Y-operation and $T \sqsubseteq S'$ a consistent specialization with respect to some X-invariant $\mathcal{I}$ with $Y \subseteq X$. Then we have $T \sqsubseteq \mu T_j.f_{\mathcal{I}}(T_j)$, where $f_{\mathcal{I}}(T_j)$ is built as in Lemma 11.

Proof. Since S' is a fixpoint we have $S' = f(S')$. T is an $\mathcal{I}$-reduced consistent specialization of S' by assumption, so the specialization

$$T \sqsubseteq f_{\mathcal{I}}(S'_{\mathcal{I}}) = f_{\mathcal{I}}\left(\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}\right)$$

follows by Lemma 11. Due to the monotonicity of $f_{\mathcal{I}}$ and because of Lemma 15 (ii) we derive further

$$f_{\mathcal{I}}\left(\left(\lim_{i\in\mathbb{N}} f^i(loop)\right)_{\mathcal{I}}\right) \sqsubseteq f_{\mathcal{I}}\left(\underbrace{\lim_{i\in\mathbb{N}} \overbrace{\left(f^i(loop)\right)_{\mathcal{I}}}^{T_{1i}}}_{T_1}\right) \quad .$$

We set $T_{2i} = f^i_{\mathcal{I}}(loop)$ and show $T_{1i} \sqsubseteq T_{2i}$ for all $i \in \mathbb{N}$ by induction. The case $i = 0$ gives $T_{10} = loop_{\mathcal{I}} = loop = T_{20}$. In the case $i > 0$ we can assume $T_{1j} \sqsubseteq T_{2j}$ for all $j < i$. T_{1i} is an $\mathcal{I}$-consistent specialization of $f^i(loop) = f(f^{i-1}(loop))$, hence we conclude

$$T_{1i} \sqsubseteq f_{\mathcal{I}}\left(\left(f^{i-1}(loop)\right)_{\mathcal{I}}\right) = f_{\mathcal{I}}\left(T_{1(i-1)}\right) .$$

by Lemma 11. Now, we apply the induction hypothesis and the monotonicity of $f_{\mathcal{I}}$ in order to obtain $f_{\mathcal{I}}\left(T_{1(i-1)}\right) \sqsubseteq f_{\mathcal{I}}\left(T_{2(i-1)}\right) = T_{2i}$, i.e. together $T_{1i} \sqsubseteq T_{2i}$ as asserted.
For $T_2 = \lim_{i\in\mathbb{N}} f^i_{\mathcal{I}}(loop)$ follows

$$\begin{array}{lcl} wlp(T_2)(\varphi) & \Leftrightarrow & \forall i.i \in \mathbb{N} \Rightarrow wlp(T_{2i})(\varphi) \\ & \Rightarrow & \forall i.i \in \mathbb{N} \Rightarrow wlp(T_{1i})(\varphi) \\ & \Leftrightarrow & wlp(T_1)(\varphi) \end{array}$$

and

$$\begin{array}{lcl} wp(T_2)(\varphi) & \Leftrightarrow & \exists i.i \in \mathbb{N} \wedge wp(T_{2i})(\varphi) \\ & \Rightarrow & \exists i.i \in \mathbb{N} \wedge wp(T_{1i})(\varphi) \\ & \Leftrightarrow & wp(T_1)(\varphi) \quad , \end{array}$$

thus the specialization $T_1 \sqsubseteq T_2$. Finally, we receive by applying Lemma 10

$$T \sqsubseteq f_{\mathcal{I}}(T_1) \sqsubseteq f_{\mathcal{I}}(T_2) = T_2 = \mu T_j . f_{\mathcal{I}}(T_j) \quad ,$$

where we use the fact that T_2 is a fixpoint. □

References

[1] J. Bell, M. Machover. *A Course in Mathematical Logic*. North-Holland 1977.

[2] E. Börger, E. Grädel, Y. Gurevich. *The Classical Decision Problem*. Springer 1997.

[3] S. Link. *Eine Theorie der Konsistenzerzwingung auf der Basis arithmetischer Logik*. M.Sc. Thesis (in German). TU Clausthal 2000.

[4] G. Nelson. A Generalization of Dijkstra's Calculus. *ACM TOPLAS*. vol. 11 (4): 517-561. 1989.

[5] K.-D. Schewe. Fundamentals of Consistency Enforcement. In H. Jaakkola, H. Kangassalo, E. Kawaguchi (eds.). *Information Modelling and Knowledge Bases* X: 275-291. IOS Press 1999.

[6] K.-D. Schewe, B. Thalheim. Towards a Theory of Consistency Enforcement. *Acta Informatica*. vol. 36: 97-141. 1999.

[7] K.-D. Schewe, B. Thalheim. Limitations of Rule Triggering Systems for Integrity Maintenance in the Context of Transition Specifications. *Acta Cybernetica*. vol. 13: 277-304. 1998.

[8] K.-D. Schewe, B. Thalheim, J. Schmidt, I. Wetzel. Integrity Enforcement in Object Oriented Databases. In U. Lipeck, B. Thalheim (eds.). *Modelling Database Dynamics*: 174-195. Workshops in Computing. Springer 1993.

[9] B. Thalheim. *Dependencies in Relational Databases*. Teubner 1991.

Received May, 2000