Electronic Notes in Theoretical Computer Science 65 No. 6 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume65.html 19 pages

Prefix and Projection onto State in Duration Calculus

Dimitar P. Guelev [1,2]

School of Computer Science, University of Birmingham, UK
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
Sofia, Bulgaria

Dang Van Hung [3]

International Institute for Software Technology, The United Nations University
UNU/IIST, P.O.Box 3058, Macau SAR, China

Abstract

We study a new operator of *projection onto state* and the *prefix operator* in the extension μHDC of DC by quantifiers over state and a polyadic least fixed point operator. We give axioms and rules to enable deduction in the extension of μHDC by the new operators. Our axioms can be used to eliminate the new operators from formulas in a practically significant fragment of μHDC. This entails the decidability of certain subfragments of this fragment is preserved in the presence of the new operators.

Introduction

It is widely recognised that the basic operators of DC [ZHR91] such as the chop operator, the least fixed point operator and the quantifiers [Pan95], are only theoretically sufficient to specify the behaviour of real time systems. The proof rules and axioms about these operators are only theoretically adequate to manipulate the obtained specifications and do verification. In practice it often pays back to use an extended kit of basic constructs in specifications and this way achieve brevity in denoting recurring patterns of clear intuitive meaning. That is why a number of *derived operators* have been proposed

[1] D. Guelev's work on this article was partially supported through Contract No. I-1102/2001 by the Ministry of Education and Science of the Republic of Bulgaria.
[2] Email: gelevdp@math.bas.bg
[3] Email: dvh@iist.unu.edu

by various authors. The use of such operators and derived axioms and rules about them can turn crucial to keep the complexity of specification and deductive verification by DC reasonably low. Derived operators often make the correspondence between design and specification by DC simpler and more intuitive. A thoroughly studied set of such operators are e.g. the *implementables* [Rav95,Die00]

In this paper we study a new operator of *projection onto state* and the *prefix operator* in the extension of DC by quantifiers over state and a polyadic least fixed point operator known as μHDC [Gue00a]. Projection onto state was introduced to DC^* in [DVH99]. It can be regarded as a real time variant of a discrete time ITL projection operator as known from [HMM83]. This operator provides a way to reconcile the *true synchrony hypothesis*, which says that computation does not take time in real-time systems, with reality, where computation does take time, which is difficult to calculate accurately and of negligible size, but needed to keep the causal ordering of computation steps clear. By means of projection onto state requirements on concurrent real-time programs' behaviour which have been formulated without taking computation time into account and specifications of this behaviour where computation time is explicitly accounted of can be put together in μHDC formulas.

We draw attention to the prefix operator, because it allows to straightforwardly define properties of initial parts of observed behaviours. Together with the *suffix* operator, which is defined symmetrically, it allows to specify the possibility for an observed behaviour to be part of some behaviour that extends out of observation into the future and/or into the past.

Along with the definition of the two operators and a proposal of their application to the specification and verification of concurrent real time programs, the paper presents the following results: We give comprehensive lists of axioms and rules which enable deduction in the extension of μHDC by the new operators. Our axioms can be used as reduction rules which enable the elimination of the new operators from formulas which commence in specifications of the proposed kind. This entails that there is a practically significant fragment of μHDC where the prefix and projection-onto-state operators can be regarded as derived operators and the decidability of certain subfragments of this fragment is preserved in the presence of the new operators.

1 Preliminaries on Real Time μHDC

In this paper we present μHDC for the case of *real* time only. We allow real-valued state variables [ZRH93], which are used to specify data manipulation in our example specification of the behaviour of concurrent interleaving processes by DC with projection and prefix. We do not mention neighbourhood terms and some of the higher-order quantifiers of μHDC to keep our presentation concise. That is why the variant of μHDC here is closer to that of μDC from [Pan95], where μ was first introduced to DC, and HDC from [ZGZ99].

1.1 Languages

A μHDC vocabulary consists of *constant symbols* a, b, c, ..., *function symbols* f, g, ... and *relation symbols* R, ... of specified arities, *individual variables* x, y, ... and *boolean state variables* P, Q, ... and *real state variables* p, q, ... Throughout the paper we denote the various kinds of μHDC symbols by the same letters as here. We rely on the usage of these letters to implicitly indicate the kinds of the particular symbols in consideration, for the sake of brevity.

Constant symbols, function symbols and relation symbols can be either *rigid* or *flexible*. Rigid symbols are distinguished for a restriction imposed on their interpretations. Flexible relation symbols of arity 0 and flexible constant symbols are also called *temporal propositional letters* and *temporal variables* respectively. The letters X, Y, ..., are tacitly assumed to denote temporal propositional letters. Individual variables are rigid. State variables are flexible.

Every μHDC language contains the rigid constant symbol 0, the temporal variable ℓ, the rigid binary function symbol $+$, the rigid binary relation symbols $=$ and $\leq$, and infinite sets of individual variables, state variables and temporal propositional letters.

Given the vocabulary of a μHDC language, its *state terms* s, *state expressions* S, *terms* t and *formulas* φ are defined by the BNFs:

$$s ::= c|x|p|f(s,\dots,s)$$

$$S ::= \mathbf{0}|P|R(s,\dots,s)|S \Rightarrow S$$

$$t ::= c|x|\textstyle\int S|f(t,\dots,t)$$

$$\varphi ::= \bot|R(t,\dots,t)|\neg\varphi|\varphi \vee \varphi|(\varphi;\varphi)|\exists x\varphi|\exists p\varphi|\exists P\varphi|\mu_i X\dots X.\varphi,\dots,\varphi$$

Only rigid constant, function and relation symbols are allowed in state terms s and state expressions S. Formulas of the kind $\mu_i X_1 \dots X_m.\varphi_1,\dots,\varphi_n$ are well-formed only if $X_1,\dots,X_m$ are distinct variables with all their occurrences in $\varphi_1,\dots,\varphi_n$ being in the scope of an even number of negations, and $m = n$. Terms and formulas built using rigid symbols only are called *rigid*.

1.2 Semantics

An abstract model M for a μHDC language $\mathbf{L}$ is a pair $\langle F, I\rangle$, where F describes the particular structure of time in M, and I describes the meaning of $\mathbf{L}$'s non-logical symbols in M, including the variables. In this paper F is always the linearly ordered group of the reals $\langle \mathbf{R}, 0, +, \leq\rangle$. For this reason we identify models with their interpretation components I.

The auxiliary notation below helps define these interpretations concisely.

Definition 1.1 We denote the set $\{[\tau_1,\tau_2] : \tau_1,\tau_2 \in \mathbf{R}, \tau_1 \leq \tau_2\}$ by $\mathbf{I}$. Given $\sigma_1,\sigma_2 \in \mathbf{I}$, $\sigma_1;\sigma_2$ stands for $\sigma_1 \cup \sigma_2$ iff $\max\sigma_1 = \min\sigma_2$. A function f on $\mathbf{R}$

has the *finite variability property*, if its range is finite and, given $\tau_1, \tau_2 \in \mathbf{R}$, $\{\tau : f(\tau) = c \text{ and } \tau_1 \leq \tau < \tau_2\}$ is a finite union of intervals of the kind $[\tau', \tau'')$ for every c in the range of f.

The finite variability property reflects the well-known fact that $\{0,1\}$-valued signals and other program variables, which are common parts of modelled systems, change their values only finitely many times in any given bounded interval of time.

Let $\mathbf{L}$ be some μHDC language.

Definition 1.2 An μHDC *interpretation* I of $\mathbf{L}$ is a function on the set of $\mathbf{L}$'s non-logical symbols, including the variables. The types of the values of I for symbols of the various kinds are as follows:

$$I(x) \in \mathbf{R} \qquad I(f) : \mathbf{I} \times \mathbf{R}^n \to \mathbf{R} \qquad I(P) : \mathbf{R} \to \{0,1\}$$

$$I(c) : \mathbf{I} \to \mathbf{R} \quad I(R) : \mathbf{I} \times \mathbf{R}^n \to \{0,1\} \quad I(p) : \mathbf{R} \to \mathbf{R}$$

Here n stands for the arity of R and f, respectively. Interpretations of state variables should have the finite variability property. Rigid symbols' interpretations should not depend on their interval argument at all. That is why they are often treated as functions of their real arguments only, and just elements of $\mathbf{R}$ in the case of 0-ary symbols.

$I(0)$, $I(+)$, $I(\leq)$, $I(=)$ and $I(\ell)$ should be the corresponding components of $\langle \mathbf{R}, 0, +, \leq \rangle$, equality on $\mathbf{R}$ and $\lambda\sigma. \max\sigma - \min\sigma$, respectively.

Definition 1.3 Given an interpretation I of $\mathbf{L}$, the values $I_\tau(s)$ of state term s and $I_\tau(S)$ of a state expression S at time point τ, and $I_\sigma(t)$ of a term t at an interval $\sigma \in \mathbf{I}$ are defined by the clauses:

$$\begin{array}{lclllcl}
I_\tau(c) & = & I(c)([\tau,\tau]) & I_\tau(p) & & = & I(p)(\tau) \\
I_\tau(x) & = & I(x) & I_\tau(f(s_1,\ldots,s_n)) & & = & I(f)([\tau,\tau], I_\tau(s_1),\ldots,I_\tau(s_n)) \\
I_\tau(\mathbf{0}) & = & 0 & I_\tau(R(s_1,\ldots,s_n)) & & = & I(R)([\tau,\tau], I_\tau(s_1),\ldots,I_\tau(s_n)) \\
I_\tau(P) & = & I(P)(\tau) & I_\tau(S_1 \Rightarrow S_2) & & = & \max\{1 - I_\tau(S_1), I_\tau(S_2)\} \\
I_\sigma(c) & = & I(c)(\sigma) & I_\sigma(\int S) & & = & \int\limits_{\min\sigma}^{\max\sigma} I_\tau(S)d\tau \\
I_\sigma(x) & = & I(x) & I_\sigma(f(t_1,\ldots,t_n)) & & = & I(f)(\sigma, I_\sigma(t_1),\ldots,I_\sigma(t_n))
\end{array}$$

The choice of $[\tau,\tau]$ to occur in the clause about $I_\tau(c)$ is arbitrary. Only rigid c are allowed in state expressions, and such c do not depend on the reference interval for their values. The same applies to the clause about $I_\tau(R(s_1,\ldots,s_n))$.

Given a variable V of any kind, interpretations I and J of $\mathbf{L}$ are said to *V-agree*, if $I(s) = J(s)$ for all non-logical symbols $s \not\equiv V$ from $\mathbf{L}$.

Let $\chi_A : \mathbf{I} \to \{0,1\}$ stand for the characteristic (membership) function of $A \subseteq \mathbf{I}$. Let $X_1,\ldots,X_n$ be temporal propositional letters from $\mathbf{L}$. Given $A_1,\ldots,A_n \subseteq \mathbf{I}$, we introduce the interpretation $I_{X_1,\ldots,X_n}^{A_1,\ldots,A_n}$ of $\mathbf{L}$ which is defined by the equalities $I^{A_1,\dots,A_n}_{X_1,\dots,X_n}(X_i) = \chi_{A_i}$, $i = 1,\dots,n$, and $I^{A_1,\dots,A_n}_{X_1,\dots,X_n}(s) = I(s)$ for $s \notin \{X_1,\dots,X_n\}$. Let $\mu_j X_1 \dots X_n.\varphi_1,\dots,\varphi_n$ be a formula in $\mathbf{L}$. Then the mappings $F_i : \left(2^{\mathbf{I}}\right)^n \to 2^{\mathbf{I}}$ that we define by the equalities:

$$F_i(A_1,\dots,A_n) = \{\sigma \in \mathbf{I} : I^{A_1,\dots,A_n}_{X_1,\dots,X_n}, \sigma \models \varphi_i\}, \quad i = 1,\dots,n,$$

are monotonic, and consequently the system of equations:

$$F_i(A_1,\dots,A_n) = A_i, \quad i = 1,\dots,n$$

has a least solution with respect to $A_1,\dots,A_n$, relative to the $\subseteq$ ordering relation. We denote the components of this solution, as they appear in their standard ordering, by $A_{I,\mu_1 X_1 \dots X_n.\varphi_1,\dots,\varphi_n},\dots,A_{I,\mu_n X_1 \dots X_n.\varphi_1,\dots,\varphi_n}$.

Definition 1.4 The modelling relation $\models$ is defined on interpretations I of $\mathbf{L}$, intervals $\sigma \in \mathbf{I}$ and formulas φ from $\mathbf{L}$ by the clauses:

$$
\begin{array}{ll}
I,\sigma \not\models \bot & \\
I,\sigma \models R(t_1,\dots,t_n) & \text{iff } I(R)(\sigma, I_\sigma(t_1),\dots,I_\sigma(t_n)) = 1 \\
I,\sigma \models \neg\varphi & \text{iff } I,\sigma \not\models \varphi \\
I,\sigma \models \varphi \vee \psi & \text{iff either } I,\sigma \models \psi \text{ or } I,\sigma \models \varphi \\
I,\sigma \models (\varphi;\psi) & \text{iff there exist } \sigma_1,\sigma_2 \in \mathbf{I} \text{ such that} \\
 & \sigma = \sigma_1;\sigma_2,\ I,\sigma_1 \models \varphi \text{ and } I,\sigma_2 \models \psi \\
I,\sigma \models \exists V \varphi & \text{iff } J,\sigma \models \varphi \text{ for some } J \text{ which } V\text{-agrees with } I \\
I,\sigma \models \mu_i X_1 \dots X_n.\varphi_1,\dots,\varphi_n & \text{iff } \sigma \in A_{I,\mu_i X_1 \dots X_n.\varphi_1,\dots,\varphi_n}
\end{array}
$$

1.3 Abbreviations

We use $\top$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\neq$, $\geq$, $<$, $>$ as abbreviations and infix notation in the usual way. The following abbreviations are *DC*-specific:

$\mathbf{1} \rightleftharpoons \mathbf{0} \Rightarrow \mathbf{0}$, $\lceil S \rceil \rightleftharpoons \int S = \ell \wedge \ell \neq 0$, $\Diamond\varphi \rightleftharpoons ((\top;\varphi);\top)$, $\Box\varphi \rightleftharpoons \neg\Diamond\neg\varphi$.

Iteration $(.)^*$ and *positive iteration* $(.)^+$ can be defined in μHDC by the clauses

$\varphi^* \rightleftharpoons \mu X.\ell = 0 \vee (\varphi;X)$, $\varphi^+ \rightleftharpoons (\varphi;\varphi^*)$.

The state variable quantifier enables the specification of hiding of local variables [ZGZ99,HX99]. Another use of this quantifier is to express *super-dense chop* [ZH96,HX99].

The least fixed point operator in *DC* enables the straightforward specification of recursive invocations in temporal programs. Assume that the temporal propositional letter X is used to denote a complete execution of some recursive temporal procedure. Then the behaviour of this procedure can be described by a formula φ which has occurrences of X to denote recursive self-invocations. Finally, a closed form of the specification of this behaviour can be given by the formula $\mu_1 X.\varphi$. The μHDC polyadic form of μ can similarly be used to specify the behaviour of collections of mutually recursive procedures, as shown in Section 3 below. Polyadic μ has the same expressive power as unary μ. A polyadic μ formula can be reduced to an equivalent unary μ one using the *Bekič principle* (cf. e.g. [AN01]). For instance,

$\models_{\mu HDC} \mu_1 X_1 X_2.\varphi_1, \varphi_2 \Leftrightarrow \mu X_1.[\mu X_2.\varphi_2/X_2]\varphi_1$.

Polyadic μ seems more convenient than unary μ with nested occurrences of for the purposes of this paper.

2 Definitions of Projection onto State and Prefix

2.1 Projection onto State

The projection (φ/S) of formula φ onto state expression S holds at interval σ under interpretation I, if φ holds at the interval $\sigma^{\lambda\tau.I_\tau(S)}$ under the interpretation $I^{\lambda\tau.I_\tau(S)}$. The interval $\sigma^{\lambda\tau.I_\tau(S)}$ is obtained by gluing the parts of σ that satisfy S. $I^{\lambda\tau.I_\tau(S)}$ is obtained by transferring the correspondence of (truth) values of symbols under I from time points and subintervals of σ to their images in $\sigma^{\lambda\tau.I_\tau(S)}$. This definition can be made precise in several ways. Below we give our choice of doing so.

Let the syntax for μHDC formulas be extended to allow formulas of the kind (φ/S). We use the auxiliary notation below to extend the relation $\models$ to formulas of this kind.

Let $h : \mathbf{R} \to \{0, 1\}$ have the finite variability property, and $\delta_h : \mathbf{R} \to \mathbf{R}$ be defined by the equality

$$\delta_h(\tau) = \int\limits_0^\tau h(\tau')d\tau'.$$

Let $\Sigma_h = \{\delta_h(\tau) : \tau \in \mathbf{R}\}$. By δ_h, the collection of intervals $\{\tau \in \mathbf{R} : h(\tau) = 1\}$ is glued into a single interval Σ_h. Clearly Σ_h is either a closed interval, or a semiclosed unbounded interval, or the entire $\mathbf{R}$, and $0 \in \Sigma_h$.

To transfer arbitrary interpretations from $\mathbf{R}$ to Σ_h as embedded in $\mathbf{R}$, we need a converse of δ. Let $\delta_h^{-1} : \mathbf{R} \to 2^{\mathbf{R}}$ be the *multiple-valued* converse of δ_h, which is defined by the equality

$$\delta_h^{-1}(\tau') = \{\tau \in \mathbf{R} : \delta_h(\tau) = \tau'\}.$$

We need a monotonic extension of a single-valued branch of δ_h^{-1} to $\mathbf{R}$. The extension γ_h with this property we choose to employ can be defined as follows:

$$\gamma_h(\tau') = \begin{cases} \tau' - \inf \Sigma_h + \max \delta_h^{-1}(\inf \Sigma_h) & \text{if } \tau' < \inf \Sigma_h < \sup \Sigma_h; \\ \max \delta_h^{-1}(\tau') & \text{if } \inf \Sigma \leq \tau' < \sup \Sigma_h; \\ \tau' - \sup \Sigma_h + \min \delta_h^{-1}(\sup \Sigma_h) & \text{if } \inf \Sigma_h < \sup \Sigma_h \leq \tau'; \\ 0 & \text{if } \inf \Sigma_h = \tau' = \sup \Sigma_h = 0. \end{cases}$$

Note that the cases above depend on the kind of interval Σ_h is and not just on τ'. The reader should retrace the definition with the various possibilities for Σ_h in mind, to get used to it.

Definition 2.1 Given an interpretation I of some μHDC language $\mathbf{L}$, the *projection* I^h *of* I *onto (the support of)* h is the μHDC interpretation of $\mathbf{L}$ which is defined by the equalities:

$$
\begin{array}{lcl}
I^h(x) & = & I(x) \text{ for individual variables } x \\
I^h(c)(\sigma) & = & I(c)([\gamma_h(\min\sigma), \gamma_h(\max\sigma)]) \text{ for constants } c \not\equiv \ell \\
I^h(s)(\sigma, d_1, \dots, d_n) & = & I(s)([\gamma_h(\min\sigma), \gamma_h(\max\sigma)], d_1, \dots, d_n) \\
 & & \text{for } n\text{-ary function and relation symbols } s \\
I^h(P)(\tau) & = & I(P)(\gamma_h(\tau)) \text{ for state variables } P
\end{array}
$$

Given $\sigma \in \mathbf{I}$, the *projection* σ^h *of* σ *onto (the support of)* h is $[\delta_h(\min\sigma), \delta_h(\max\sigma)]$.

Now let φ be a formula and H be a state expression in $\mathbf{L}$, respectively. Let $h = \lambda\tau.I_\tau(H)$. Then

$I, \sigma \models (\varphi/H)$ iff $I^h, \sigma^h \models \varphi$.

Note that with γ_h defined and used as above, I^h is obtained from I by clipping off parts of $\mathbf{R}$ which are surrounded by parts where h evaluates to 1 only. In case Σ_h is (semi)bounded, that is, if $\inf\Sigma_h \in \mathbf{R}$, or $\sup\Sigma_h \in \mathbf{R}$, or both, the values of I on $\{\tau \in \mathbf{R} : \tau < \inf\Sigma_h\}$ and $\{\tau \in \mathbf{R} : \tau \geq \sup\Sigma_h\}$ are transferred to I^h with no loss.

2.2 Prefix and Suffix

Informally, $I, \sigma \models \mathsf{pref}(\varphi)$, if some extension of the restriction of I to σ into the future satisfies φ at a possibly longer interval beginning at the same time point as σ. In the case of suff the extension is sought into the past.

Definition 2.2 Given $\sigma \in \mathbf{I}(\mathbf{R})$, interpretations I_1 and I_2 of μHDC language $\mathbf{L}$ σ*-agree*, if

$I_1(x) = I_2(x)$	for individual variables x
$I_1(c)(\sigma') = I_2(c)(\sigma')$	for constants c, if $\sigma' \subseteq \sigma$;
$I_1(s)(\sigma', d_1, \dots, d_n) = I_2(s)(\sigma', d_1, \dots, d_n)$	for n-ary function and relation symbols s, if $\sigma' \subseteq \sigma$ and $d_1, \dots, d_n \in \mathbf{R}$;
$I_1(P)(\tau') = I_2(P)(\tau')$	for state variables P from $\mathbf{L}$ and $\min\sigma \leq \tau' < \max\sigma$.

The proposition below explains σ-agreeing:

Proposition 2.3 *Let* $\sigma, \sigma' \in \mathbf{I}$ *and* $\sigma' \subseteq \sigma$. *Let* I_1 *and* I_2 *be interpretations of* $\mathbf{L}$ *which* σ*-agree, and* φ *be a formula in* $\mathbf{L}$. *Then* $I_1, \sigma' \models \varphi$ *is equivalent to* $I_2, \sigma' \models \varphi$.

We define the unary modal operators pref and suff by the clauses:

$I, \sigma \models \mathsf{pref}(\varphi)$ iff $I', \sigma' \models \varphi$ for some I' and σ' such that

I' σ-agrees with I, $\sigma' \supseteq \sigma$ and $\min \sigma' = \min \sigma$.

$I, \sigma \models \mathsf{suff}(\varphi)$ iff $I', \sigma' \models \varphi$ for some I' and σ' such that

I' σ-agrees with I, $\sigma' \supseteq \sigma$ and $\max \sigma' = \max \sigma$.

3 Specification by DC with Projection and Prefix

In this section we show how the operators pref and $(./.)$ can be used to specify the behaviour of interleaving processes and requirements on such behaviour. We consider real-time programs $\mathbf{P}$ of the kind

$P_1 || \ldots || P_n$,

where $P_1, \ldots, P_n$ are $\mathbf{P}$'s component processes, which run concurrently. The syntax of individual component processes P is described by the BNF

$$P ::= \mathtt{skip} | x := e | X | \mathtt{delay}\ r | \mathtt{await}\ b | (P; P) | \mathtt{if}\ b\ \mathtt{then}\ P\ \mathtt{else}\ P |$$

$$\mathtt{letrec}\ P\ \mathtt{where}\ X : P; \ldots X : P;$$

where x stands for a variable, e,r and b stand for expressions of the appropriate types and are built using variables, constants and operations (e.g. arithmetic operations,) and X stands for a subprocess name in `letrec`. Subprocess name X may occur in process P only if P is in the scope of a `letrec` statement which binds X. We assume that real valued expressions have the syntax of real μHDC state terms, and boolean valued expressions have the syntax of μHDC state expressions, for the sake of simplicity.

The statements which appear in the above BNF are executed as follows:

`skip` Do nothing.

$x := e$ Evaluate e and then set x to the value of e.

`delay` r Evaluate r and postpone all subsequent action of the relevant process by the obtained number of time units.

`await` b Wait until b becomes true and terminate. In case b never becomes true, `await` b never terminates.

$(P_1; P_2)$ Execute P_1 first, and, in case P_1 terminates, execute P_2.

`if` b `then` P_1 `else` P_2 Evaluate b first. If b is true, then execute P_1. Otherwise execute P_2.

X Execute the subprocess labelled X from the innermost running `letrec` statement which binds X.

`letrec` P `where` $X_1 : P_1; \ldots X_n : P_n$; Execute P with this statement being the innermost running `letrec` statement which binds $X_1, \ldots, X_n$.

We denote the set of variables which occur in process P_i by $Var(P_i)$. We specify the behaviour of $\mathbf{P}$ by μHDC formulas in the μHDC language $\mathbf{L}(\mathbf{P})$

with the following non-logical symbols:

A state variable x for every $x \in \bigcup_{i=1}^{n} Var(P_i)$.

Rigid symbols of the appropriate kinds and arities and the same names for all constants, functions and relations which occur in boolean and real-valued expressions in $\mathbf{P}$.

We assume that boolean variable x from $\mathbf{P}$ are represented by boolean state variables, and real-valued variables are represented by real state variables.

The boolean state variables R_i and W_i, $i = 1, \dots, n$. $\lceil R_i \rceil$ indicates that P_i performs computation and therefore has exclusive access to the variables from $Var(P_i)$ during the reference interval. W_i indicates that P_i has terminated.

The boolean state variable N. $\lceil N \rceil$ indicates that the reference interval consists of *negligible time.* N has a key role in our approach to handling the *true synchrony hypothesis* by *DC* with projection onto state. According to this hypothesis, computation consumes no time, and only awaiting external synchronisation and explicitly stated delays consume time. We describe behaviours of real time programs by taking into account that in fact computation *does take* time. However, this time can be regarded as *negligible* and marked by N. The key observation in our approach is that models of *DC* which describe behaviours of a program $\mathbf{P}$, should satisfy $(\varphi/\neg N)$, provided that φ is a *DC* formula which specifies some property of the behaviours of $\mathbf{P}$ under the true synchrony hypothesis.

Propositional temporal letters X and X' for each subprocess name X which occurs in a `letrec` statement in $\mathbf{P}$.

The kinds of non-logical symbols which are obligatory for μHDC languages in general, in particular, an individual variable u.

Given $\mathbf{L}(\mathbf{P})$, we introduce a formula $\mathcal{A}$ in $\mathbf{L}(\mathbf{P})$ which specifies the general conditions of running $\mathbf{P}$. For each individual subprocess P of P_i, $i = 1, \dots, n$, we introduce two μHDC formulas $[\![P]\!]_i$ and $[\![P]\!]'_i$. Given a model M of $\mathbf{L}(\mathbf{P})$ and interval $\sigma \in \mathbf{I}$, we define $[\![P]\!]_i$ and $[\![P]\!]'_i$ so that the following connection between them and the behaviour of P holds:
$I, \sigma \models \Box\mathcal{A} \wedge [\![P]\!]_i$ iff I describes a complete finite run of P in σ.
I describes a non-terminating run of P starting at τ_0 iff $I, [\tau_0, \tau] \models \Box\mathcal{A} \wedge [\![P]\!]'_i$ for all $\tau \geq \tau_0$.

To define $\mathcal{A}$, $[\![P]\!]_i$ and $[\![P]\!]'_i$ concisely, we use some abbreviations. Let $V \subseteq Var(P_i)$. The formula

$$\mathsf{K}_i(V) \rightleftharpoons \bigwedge_{x \in Var(P_i) \setminus V} \exists u(\textstyle\int (x = u) = \ell)$$

together with some conditions introduced below, says that P_i variables, except eventually the ones from V, change their values neither within the reference interval, nor at its end. The formula

$$\mathsf{S}_i(S) \rightleftharpoons (\lceil N \wedge \neg R_i \rceil ; \mathsf{K}_i(\emptyset) \wedge \lceil R_i \wedge S \rceil)$$

says that the reference interval consists of two non-zero-length parts. In the second part, neither a P_i variable changes its value, nor its value gets accessed by some other process. The parameter S holds place for a state expression of how P_i accesses variables in this part. The first part is inserted to allow interleaving. $\mathcal{A}$ is the conjunction of the following formulas:

$\mathcal{A}_0$	A rigid formula giving the relevant algebraic properties of constants, operations and predicates which occurs in expressions e, r, b.
$\lceil\neg N\rceil \Rightarrow \bigwedge\limits_{i=1}^{n} \mathsf{K}_i(\emptyset)$	Variables get updated during computation time only.
$\bigwedge\limits_{\substack{i=1,\dots,n\\ x\in Var(P_i)}} \forall u\neg(\lceil x=u\wedge R_i\rceil;\lceil x\neq u\wedge\neg R_i\rceil)$	Each update takes place *inside* the computation
$\bigwedge\limits_{\substack{i=1,\dots,n\\ x\in Var(P_i)}} \forall u\neg(\lceil x=u\wedge \neg R_i\rceil;\lceil x\neq u\wedge R_i\rceil)$	time of some process (where R_i is true).
$\bigwedge\limits_{1\leq i<j\leq n} \int(R_i\wedge R_j)=0$	No two processes perform computation at the same time.
$\int(N\Leftrightarrow \bigwedge\limits_{i=1}^{n} W_i \vee \bigvee\limits_{i=1}^{n} R_i)=\ell$	A part of the behaviour of $\mathbf{P}$ is negligible iff some of the component processes is accessing its variables, that is, doing negligible time computation, or all the processes have terminated.
$\bigwedge\limits_{i=1}^{n} \neg(\lceil W_i\rceil;\lceil\neg W_i\rceil)$	Once P_i terminates, it stays terminated.
$\bigwedge\limits_{i=1}^{n} \neg(\lceil W_i\wedge R_i\rceil)$	Terminated processes do not access their variables.

Given $i\in\{1,\dots,n\}$ and subprocess P of P_i, $[\![P]\!]_i$ and $[\![P]\!]'_i$ are defined by the clauses:

$$
\begin{array}{lcl}
[\![\mathtt{skip}]\!]_i & \rightleftharpoons & \lceil N\wedge\neg R_i\rceil \\
[\![x:=e]\!]_i & \rightleftharpoons & \lceil N\rceil\wedge(\lceil\neg R_i\rceil;\lceil R_i\rceil\wedge\mathsf{K}_i(\{x\})\wedge \\
 & & \exists u(\lceil u=e\rceil;\lceil x=u\rceil);\lceil\neg R_i\rceil) \\
[\![X]\!]_i & \rightleftharpoons & X \\
[\![\mathtt{delay}\ r]\!]_i & \rightleftharpoons & \exists u((\mathsf{S}_i(u=r);\lceil\neg R_i\rceil)\wedge u=\int\neg N) \\
[\![\mathtt{await}\ b]\!]_i & \rightleftharpoons & (\int\neg(R_i\vee b)=\ell;\mathsf{K}_i(\emptyset)\wedge\lceil N\wedge R_i\wedge b\rceil;\lceil N\wedge\neg R_i\rceil) \\
[\![(P_1;P_2)]\!]_i & \rightleftharpoons & ([\![P_1]\!]_i;[\![P_2]\!]_i) \\
[\![\mathtt{if}\ b\ \mathtt{then}\ P_1\ \mathtt{else}\ P_2]\!]_i & \rightleftharpoons & (\mathsf{S}_i(b);[\![P_1]\!]_i)\vee(\mathsf{S}_i(\neg b);[\![P_2]\!]_i) \\
[\![\mathtt{letrec}\ P\ \mathtt{where} & \rightleftharpoons & \mu_{n+1}X_1\dots X_nY.[\![P_1]\!]_i,\dots,[\![P_n]\!]_i,[\![P]\!]_i \\
\quad X_1:P_1;\dots;X_n:P_n]\!]_i & &
\end{array}
$$

$$\begin{array}{llllll} [\![\texttt{skip}]\!]'_i & \rightleftharpoons & \mathsf{pref}([\![\texttt{skip}]\!]_i) & [\![\texttt{delay}\ r]\!]'_i & \rightleftharpoons & \mathsf{pref}([\![\texttt{delay}\ r]\!]_i) \\ [\![x := e]\!]'_i & \rightleftharpoons & \mathsf{pref}([\![x := e]\!]_i) & [\![\texttt{await}\ b]\!]'_i & \rightleftharpoons & \int \neg(R_i \vee b) = \ell \\ [\![X]\!]'_i & \rightleftharpoons & X' & [\![(P_1; P_2)]\!]'_i & \rightleftharpoons & [\![P_1]\!]'_i \vee ([\![P_1]\!]_i; [\![P_2]\!]'_i) \end{array}$$

$$[\![\texttt{if}\ b\ \texttt{then}\ P_1\ \texttt{else}\ P_2]\!]'_i \rightleftharpoons \left(\begin{array}{l} \mathsf{pref}(\mathsf{S}_i(b)) \vee (\mathsf{S}_i(b); [\![P_1]\!]'_i) \vee \\ \mathsf{pref}(\mathsf{S}_i(\neg b)) \vee (\mathsf{S}_i(\neg b); [\![P_2]\!]'_i) \end{array}\right)$$

$$[\![\texttt{letrec}\ P\ \texttt{where} X_1 : P_1; \ldots ; X_n : P_n]\!]'_i \rightleftharpoons$$

$$\mu_{2n+1} X_1 \ldots X_n X'_1 \ldots X'_n Y. [\![P_1]\!]_i, \ldots , [\![P_n]\!]_i, [\![P_1]\!]'_i, \ldots , [\![P_n]\!]'_i, [\![P]\!]'_i$$

Using $[\![P_i]\!]_i$ and $[\![P_i]\!]'_i$, $i = 1, \ldots , n$, terminating runs of $\mathbf{P}$ can be specified by the formula

$$[\![\mathbf{P}]\!] \rightleftharpoons \Box\mathcal{A} \wedge \bigwedge_{i=1}^{n} ([\![P_i]\!]_i; \lceil W_i \rceil)$$

and initial subintervals of non-terminating runs of $\mathbf{P}$ of sufficient duration satisfy the formula

$$[\![\mathbf{P}]\!]' \rightleftharpoons \Box\mathcal{A} \wedge \bigvee_{J \subseteq \{1,\ldots,n\}, J \neq \emptyset} \left(\bigwedge_{i \in J} [\![P_i]\!]'_i \wedge \bigwedge_{i \in \{1,\ldots,n\} \setminus J} ([\![P_i]\!]_i; \lceil W_i \rceil) \right).$$

The projection operator allows to put down properties of the behaviours specified in the above way without keeping in mind that computation time is taken into account in the specification of these behaviours. Given an interpretation I of $\mathbf{L}(\mathbf{P})$ which describes a behaviour of $\mathbf{P}$ in some interval $\sigma \in \mathbf{I}$ with computation time taken into account, that is, with N and R_i, $i = 1, \ldots , n$, becoming 1 here and there, $I^{\lambda\tau.I_\tau(\neg N)}$ describes the same behaviour of $\mathbf{P}$ in the interval $\sigma^{\lambda\tau.I_\tau(\neg N)}$ under the true synchrony hypothesis, that is, with the computation time clipped off. Hence, if a requirement φ on this behaviour has been written without accounting of computation time, then the behaviour will satisfy φ iff $I, \sigma \models (\varphi/\neg N)$. Hence a requirement φ is generally satisfied by the terminating runs of $\mathbf{P}$ iff

$$\models_{\mu HDC} [\![\mathbf{P}]\!] \Rightarrow (\varphi/\neg N)$$

and φ is satisfied by the initial subintervals of non-terminating runs of $\mathbf{P}$ iff

$$\models_{\mu HDC} [\![\mathbf{P}]\!]' \Rightarrow \neg(\neg(\varphi/\neg N); \top)$$

Similarly, projections of the kind $(./R_i \vee \neg N)$ and $(./ \bigvee_{j \in J} R_j \vee \neg N)$ can be used to specify properties of behaviours of the entire program $\mathbf{P}$ as observable by individual component process P_i or a subset $\{P_j : j \in J\}$ of the component processes, respectively.

4 Axioms and Rules for Projection onto State

In this section we study projection onto state as one of the operators of μHDC. We formulate some interesting properties of $(./.)$ as μHDC valid formulas and proof rules. We specify a fragment of μHDC with projection onto state which admits a simple truth preserving translation into μHDC without projection. This translation can be defined by taking some of our axioms as the translation rules. The existence of the translation entails a decidability result about another smaller fragment of μHDC with projection. Finally, we give a general proof rule about projection.

4.1 Projection onto State and Basic HDC Operators

(1) $\models_{\mu HDC} \varphi \Leftrightarrow (\varphi/H)$ for rigid φ from $\mathbf{L}$.

Let $\sigma \in \mathbf{I}$, I be an interpretation of some μHDC language $\mathbf{L}$, H be a state expression in $\mathbf{L}$ and $h = \lambda\tau.I_\tau(H)$. Then $\min\sigma^h \leq \tau < \max\sigma^h$ implies $I^h_\tau(H) = 1$. Hence

(2) $\models_{\mu HDC} (\ell = \int H/H)$.

Since $\max\sigma^h - \min\sigma^h = \int\limits_{\min\sigma}^{\max\sigma} I_\tau(H)d\tau$, $I^h_\sigma(\ell) = I_\sigma(\int H)$. This entails that

(3) $\models_{\mu HDC} (\ell = x/H) \Leftrightarrow \int H = x$

Similar considerations show that

(4) $\models_{\mu HDC} (\int S = x/H) \Leftrightarrow \int(S \wedge H) = x$.

This means that $(./H)$ can be eliminated from (φ/H) in the case of atomic φ with rigid symbols and $\int$ subterms only by putting $\int(S \wedge H)$ wherever $\int S$ occurs. $(./H)$ can be eliminated in case φ is an atomic flexible formula $R(t_1, \dots, t_n)$ which satisfies

$$\exists x(\int H = x \wedge (\Box(\int H = x \Rightarrow R(t_1, \dots, t_n)) \vee \Box(\int H = x \Rightarrow \neg R(t_1, \dots, t_n)))$$

Given this,

$$\models_{\mu HDC} R(t_1, \dots, t_n) \Leftrightarrow (R(t_1, \dots, t_n)/H)$$

If neither $\int$, nor ℓ occur in $t_1, \dots, t_n$, then

$$(5) \quad \models_{\mu HDC} \begin{array}{c} (\lceil H \rceil; \top) \wedge (\varepsilon R(t_1, \dots, t_n) \wedge \int H = x; \lceil H \rceil; \top) \Rightarrow \\ ((\varepsilon R(t_1, \dots, t_n) \wedge \ell = x; \top)/H) \end{array}$$

where ε stands for either $\neg$ or nothing. Straightforward arguments show that:

(6) $\models_{\mu HDC} (\neg\varphi/H) \Leftrightarrow \neg(\varphi/H)$

(7) $\models_{\mu HDC} (\varphi \vee \psi/H) \Leftrightarrow (\varphi/H) \vee (\psi/H)$

(8) $\models_{\mu HDC} ((\varphi; \psi)/H) \Leftrightarrow ((\varphi/H); (\psi/H))$

(9) $\models_{\mu HDC} (\exists V\varphi/H) \Leftrightarrow \exists V(\varphi/H)$, if variable V does not occur in H

(10) $\models_{\mu HDC} ((\varphi/S)/H) \Leftrightarrow (\varphi/S \wedge H)$

(11) $\models_{\mu HDC} \varphi \Rightarrow \psi$ implies $\models_{\mu HDC} (\varphi/H) \Rightarrow (\psi/H)$

(12) $\models_{\mu HDC} \int(H_1 \Leftrightarrow H_2) = \ell$ implies $\models_{\mu HDC} (\varphi/H_1) \Leftrightarrow (\varphi/H_2)$

(13) $\models_{\mu HDC} (\varphi/\mathbf{1}) \Leftrightarrow \varphi$

(14) $\models_{\mu HDC} ((\varphi; \psi)/\mathbf{0}) \Leftrightarrow (\varphi \wedge \psi/\mathbf{0})$

4.2 Projection onto State and μ

The valid formulas listed so far are sufficient to deal with $(./.)$ in HDC without μ by, e.g., driving it towards atomic formulas. Next we extend this approach a fragment of μHDC which properly contains HDC.

Proposition 4.1 *Let H be a state expression, $\varphi \rightleftharpoons \mu_i X_1 \dots X_n.\varphi_1, \dots, \varphi_n$ and none of the occurrences of $X_1, \dots, X_n$ in $\varphi_1, \dots, \varphi_n$ be in the scope of negation, nor in the scope of μ or $(./.)$. Let ψ_j be obtained from (φ_j/H) by driving projection inwards and finally replacing $(X_1/H), \dots, (X_n/H)$ by $X_1, \dots, X_n$, respectively, $j = 1, \dots, n$. Let $\psi \rightleftharpoons \mu_i X_1 \dots X_n.\psi_1, \dots, \psi_n$ Then $\models_{\mu HDC} (\varphi/H) \Leftrightarrow \psi$*

Proof. Consider the finite sets of HDC formulas $\Phi_1^k, \dots, \Phi_n^k$, $k < \omega$, which are defined by putting:

$\Phi_j^0 = \{\bot\}$

$\Phi_j^{k+1} = \Phi_j^k \cup \{[\alpha_1/X_1, \dots, \alpha_n/X_n]\varphi_i : \alpha_1 \in \Phi_1^k, \dots, \alpha_n \in \Phi_n^k\}$, $j = 1, \dots, n$

Let $\Psi_1^k, \dots, \Psi_n^k$, $k < \omega$, be defined similarly, yet using $\psi_1, \dots, \psi_n$ instead of $\varphi_1, \dots, \varphi_n$. Let $(\Phi_j^k/H) = \{(\alpha/H) : \alpha \in \Phi_j^k\}$, $j = 1, \dots, n$, $k < \omega$. It can be shown that every formula from (Φ_j^k/H) has an equivalent one in Ψ_j^k and vice versa.

The restrictions on the use of $\neg$ in $\varphi_1, \dots, \varphi_n$ entail that $I, \sigma \models \varphi$ iff $\exists k < \omega \exists \alpha \in \Phi_i^k(I, \sigma \models \alpha)$. Hence $I, \sigma \models (\varphi/H)$ is equivalent to $\exists k < \omega \exists \alpha \in \Phi_i^k(I, \sigma \models (\alpha/H))$. The latter is equivalent to $\exists k < \omega \exists \beta \in \Psi_i^k(I, \sigma \models \beta)$, that is to $I, \sigma \models \psi$. □

Apparently, the most useful corollary to this proposition is:

(16) $\models_{\mu HDC} (\varphi^*/H) \Leftrightarrow ((\varphi/H)^*; \int H = 0)$, $\qquad \models_{\mu HDC} (\varphi^+/H) \Leftrightarrow (\varphi/H)^+$

4.3 Projection onto State in General

We conclude this section with a general proof rule about $(./.)$. It applies to virtually all conservative extensions of DC with $(./.)$ with introspective modalities only. We first give some observations that suggest this rule.

Assume the notation introduced to define $(./.)$. Since the duration of σ^h never exceeds that of σ, the condition $I^h, \sigma^h \models \varphi$ can be replaced by an equivalent one of the kind $I', \sigma^{h'} \models \varphi$, where $\sigma^{h'} = [\min \sigma, \min \sigma + \max \sigma^h - \max \sigma^h]$ and I' is an interpretation of $\mathbf{L}$ that behaves on $\sigma^{h'}$ in the way I^h does on σ^h. Next, all the flexible symbols which occur in φ can be replaced by fresh ones, thus obtaining an isomorphic formula φ', and I' can be replaced by an interpretation I'' which is defined on these symbols only and yields the same values on them as I' does on the original symbols of φ. This allows the condition $I^h, \sigma^h \models \varphi$ to be replaced by $I'', \sigma^{h'} \models \varphi'$. The latter is equivalent to

$$I'', \sigma \models \exists x(\int H = x \wedge (\ell = x \wedge \varphi'; \top)),$$

provided x has no free occurrence in φ. Now let $\mathbf{L}'$ be the extension of $\mathbf{L}$ by the fresh flexible symbols used to place in φ'. Then $I \cup I''$ is an interpretation of $\mathbf{L}'$ and

$$I \cup I'', \sigma \models (\varphi/H) \Leftrightarrow \exists x(\textstyle\int H = x \wedge (\ell = x \wedge \varphi'; \top))$$

Let us specify the desired relationship between the values of I on the flexible symbols of φ at the subintervals of σ and the values of I'' on their counterparts from φ' at the corresponding subintervals of $\sigma^{h'}$ by DC formulas. Let R and R' be n-place relation symbols. We denote the formula

$$\forall x_1 \ldots \forall x_n \forall y \forall z \left(\begin{array}{l} y + z \leq \int H \Rightarrow \\ \left(\begin{array}{l} (\int H = y; (\lceil H \rceil; \top) \wedge \int H = z \wedge R(x_1, \ldots, x_n); (\lceil H \rceil; \top))) \\ \Leftrightarrow (\ell = y; R'(x_1, \ldots, x_n) \wedge \ell = z; \top) \end{array} \right) \end{array} \right)$$

by $R \sim_H R'$. Let $n \geq 2$ and f and f' be $n-1$-ary function symbols. Then we denote formula that is obtained by putting $f(x_1, \ldots, x_{n-1}) = x_n$ and $f'(x_1, \ldots, x_{n-1}) = x_n$ in place of $R(x_1, \ldots, x_n)$ and $R'(x_1, \ldots, x_n)$ respectively by $f \sim_H f'$. For flexible constants c and c', $c \sim_H c'$ is the result of substituting $c = x_1$ and $c' = x_1$ in the respective places in the above formula with $n = 1$. The subformula $(\lceil H \rceil; \top)$ in $R \sim_H R'$ is to ascertain that the restriction of I to the subintervals of σ bears enough information to define I'' to the subintervals of $\sigma^{h'}$, because it is possible to have $\gamma_h(\delta_h(\tau)) > \tau$, in case $h(\tau) = 0$. For boolean state variables P and P', $P \sim_H P'$ is

$$\forall x \forall y \forall z \left(y + z \leq \int H \Rightarrow \left(\begin{array}{l} (\int H = y; \int H = z \wedge \int (P \wedge H) = x; \top) \\ \Leftrightarrow (\ell = y; \ell = z \wedge \int P' = x; \top) \end{array} \right) \right)$$

Finally , for real state variables p and p', $p \sim_H p'$ is

$$\forall x \forall y \forall z \forall t \left(y + z \leq \int H \Rightarrow \left(\begin{array}{l} (\int H = y; \int H = z \wedge \int (p = t \wedge H) = x; \top) \\ \Leftrightarrow (\ell = y; \ell = z \wedge \int (p' = t) = x; \top) \end{array} \right) \right)$$

Now, given that $s_1, \ldots, s_n$ are the flexible symbols of φ, and $s'_1, \ldots, s'_n$ are fresh symbols of the same kinds and arities as $s_1, \ldots, s_n$, respectively, the rule can be put down as follows:

$$(PR) \quad \frac{\theta \wedge \bigwedge_{i=1}^{n} s_i \sim_H s'_i \Rightarrow (\ell = x; \ell = y \wedge [s'_1/s_1, \ldots, s'_n/s_n]\varphi; \top)}{\theta \wedge (\int H \geq x + y; \lceil H \rceil; \top) \Rightarrow \neg(\int H = x; \int H = y \wedge \neg(\varphi/H); \lceil H \rceil; \top)}$$

5 The $\lceil P \rceil$-fragment of HDC^* with Projection onto State is Decidable

The BNF for formulas in the $\lceil P \rceil$-fragment of HDC^* is

$$\varphi ::= \bot | \ell = 0 | \lceil S \rceil | \neg\varphi | \varphi \vee \varphi | (\varphi; \varphi) | \varphi^* | \exists P \varphi$$

It is known that $\exists P$ and $\neg$ can be eliminated from HDC^* $\lceil P \rceil$ formulas, that is, for every HDC^* $\lceil P \rceil$-formula an equivalent quantifier-free and negation-free one in the same vocabulary can be built. A proof in the notation of this paper can be found in [Gue00b]. The valid equivalences from Subsections 4.1 and 4.2 entail that given such a formula φ and a state expression H, an equivalent ψ to (φ/H) can be built with projection occurring in ψ only in subformulas of the kinds $(\ell = 0/H)$ and $(\lceil S\rceil/H)$. Yet the valid equivalences

$$(\ell = 0/H) \Leftrightarrow \ell = 0 \vee \lceil\neg H\rceil \text{ and } (\lceil S\rceil/H) \Leftrightarrow (\lceil H \Rightarrow S\rceil) \wedge \Diamond\lceil H\rceil$$

show that projection onto state can be eliminated from these subformulas too. Hence every formula from the $\lceil P\rceil$-fragment of HDC^* with projection can be transformed into an equivalent quantifier- and projection-free one. Validity is decidable for such formulas, as known from [ZHS93].

6 Axioms and Rules for pref and suff in μHDC

This section is structured after the previous one about $(./.)$ in μHDC.

6.1 pref, suff *and Basic HDC Operators*

The valid μHDC formulas with pref and suff below make no explicit reference to μ. Most of them deal with the interaction between pref and the HDC operators only. To obtain the corresponding equivalences about suff, one should interchange the operands of $(.;.)$.

(1) $\models_{\mu HDC} \mathsf{pref}(\varphi) \Leftrightarrow \varphi$ for rigid φ
(2) $\models_{\mu HDC} \mathsf{pref}(\int S = x) \Leftrightarrow \int S \leq x$
(3) $\models_{\mu HDC} \mathsf{pref}(R(x_1, \dots, x_n))$ for flexible R
(4) $\models_{\mu HDC} \mathsf{pref}(f(x_1, \dots, x_n) = x_{n+1})$ for flexible f
(5) $\models_{\mu HDC} \mathsf{pref}(c = x)$ for flexible $c \not\equiv \ell$
(6) $\models_{\mu HDC} \neg\mathsf{pref}(\varphi) \Rightarrow \mathsf{pref}(\neg\varphi)$
(7) $\models_{\mu HDC} \mathsf{pref}(\varphi \wedge \psi) \Rightarrow \mathsf{pref}(\varphi) \wedge \mathsf{pref}(\psi)$
(8) $\models_{\mu HDC} \mathsf{pref}(\varphi \vee \psi) \Leftrightarrow \mathsf{pref}(\varphi) \vee \mathsf{pref}(\psi)$
(9a) $\models_{\mu HDC} \mathsf{pref}((\varphi;\psi)) \Rightarrow \mathsf{pref}(\varphi) \vee (\varphi; \mathsf{pref}(\psi))$
(9b) $\models_{\mu HDC} (\varphi; \mathsf{pref}(\psi)) \Rightarrow \mathsf{pref}((\varphi;\psi))$
(9c) $\models_{\mu HDC} \ell = 0 \wedge \mathsf{suff}(\varphi) \Rightarrow \mathsf{pref}(\psi)$ implies $\models_{\mu HDC} \mathsf{pref}(\varphi) \Rightarrow \mathsf{pref}((\varphi;\psi))$
(10) $\models_{\mu HDC} \mathsf{pref}(\exists x\varphi) \Leftrightarrow \exists x\mathsf{pref}(\varphi)$
(11) $\models_{\mu HDC} \mathsf{pref}(\exists P\varphi) \Leftrightarrow \exists P\mathsf{pref}(\varphi)$
(12) $\models_{\mu HDC} \varphi \Rightarrow \psi$ implies $\models_{\mu HDC} \mathsf{pref}(\varphi) \Rightarrow \mathsf{pref}(\psi)$
(13) $\models_{\mu HDC} (\neg\mathsf{pref}(\varphi); \top) \Rightarrow \neg\varphi$
(14) $\models_{\mu HDC} \varphi \Rightarrow \neg(\neg\psi; \top)$ implies $\models_{\mu HDC} \mathsf{pref}(\varphi) \Rightarrow \psi$
(15) $\models_{\mu HDC} \mathsf{pref}(\mathsf{pref}(\varphi)) \Leftrightarrow \mathsf{pref}(\varphi)$
(16) $\models_{\mu HDC} (\mathsf{pref}(\varphi)/H) \Rightarrow \mathsf{pref}((\varphi/H))$

Superpositions of pref *and* suff

(17) $\models_{\mu HDC} \varphi \Rightarrow \Box\mathsf{pref}(\mathsf{suff}(\varphi))$
(18) $\models_{\mu HDC} \varphi \Rightarrow \Box\psi$ implies $\models_{\mu HDC} \mathsf{pref}(\mathsf{suff}(\varphi)) \Rightarrow \psi$

(19) $\models_{\mu HDC} \Diamond\varphi \Rightarrow \psi$ implies $\models_{\mu HDC} \varphi \Rightarrow \mathsf{pref}(\mathsf{suff}(\psi))$
(20) $\models_{\mu HDC} \mathsf{suff}(\mathsf{pref}(\varphi)) \Leftrightarrow \mathsf{pref}(\mathsf{suff}(\varphi))$

6.2 pref*,* suff *and* μ

Just like in the case of projection, the emphasis in the above list of valid formulas and rules about pref is to enable driving pref and suff towards atomic formulas to the possible extent. In this subsection we show that this can be done with some μ formulas too. The fragment of μHDC we take is part of that from Proposition 4.1.

Let H be a state expression, $\psi_i \rightleftharpoons \mu_i X_1 \dots X_n.\varphi_1, \dots, \varphi_n$, $i = 1, \dots, n$ and none of the free occurrences of $X_1, \dots, X_n$ in φ_j, $j = 1, \dots, n$, be in the scope of negation, nor in the scope of μ, $(./.)$ or a quantifier which binds an individual variable.

Lemma 6.1 *Each of the φ_j, $j = 1, \dots, n$, is equivalent to a disjunction of formulas of the kind $(\alpha_1; \dots; \alpha_l)$, where either $\alpha_k \in \{X_1, \dots, X_n\}$ or α_k has no free occurrences of $X_1, \dots, X_n$, $k = 1, \dots, l$.*

Proof. The equivalent formula can be obtained by applying

$$\models_{\mu HDC} \exists P(\varphi \vee \psi) \Leftrightarrow \exists P\varphi \vee \exists P\psi \text{ and } \models_{\mu HDC} \exists P(\varphi; \psi) \Leftrightarrow (\exists P\varphi; \exists P\psi)$$

and the distributivity of $(.;.)$ over disjunction as reduction rules on subformulas of φ_j which have occurrences of $X_1, \dots, X_n$. $\Box$

In the sequel we assume that

$$\varphi_j \doteq \bigvee_{k=1}^{m_j} (\alpha_{j,k,0}; X_{i_{j,k,1}}; \alpha_{j,k,0}; \dots; \alpha_{j,k,l_{j,k}}; X_{i_{j,k,l_{j,k}}}; \alpha_{j,k,l_{j,k}}), \quad i, j = 1, \dots, n$$

where $\alpha_{j,k,0}, \dots, \alpha_{j,k,l_{j,k}}$ have no free occurrences of $X_1, \dots, X_n$. Since $\models_{\mu HDC}$ $\psi_i \Leftrightarrow [\psi_1/X_1, \dots \psi_n/X_n]\varphi_i$ we have

$$\models_{\mu HDC} \mathsf{pref}(\psi_i) \Leftrightarrow \bigvee_{k=1}^{m_j} \left(\begin{array}{l} \bigvee_{p=0}^{l_{j,k}} (\alpha_{j,k,0}; \psi_{i_{j,k,1}}; \dots; \psi_{i_{j,k,p}}; \mathsf{pref}(\alpha_{j,k,p})) \vee \\ \bigvee_{p=1}^{l_{j,k}} (\alpha_{j,k,0}; \psi_{i_{j,k,1}}; \dots; \psi_{i_{j,k,p-1}}; \alpha_{j,k,p-1} \mathsf{pref}(\psi_{i_{j,k,p}})) \end{array} \right)$$

Substituting $Y_1, \dots, Y_n$ for $\mathsf{pref}(\psi_1), \dots, \mathsf{pref}(\psi_n)$ in the above equivalences suggests that

$$\models_{\mu HDC} \mathsf{pref}(\psi_i) \Leftrightarrow \mu_i Y_1 \dots Y_n.\chi_1, \dots, \chi_n, \quad i = 1, \dots, n$$

where

$$\chi_i \rightleftharpoons \bigvee_{k=1}^{m_j} \left(\begin{array}{l} \bigvee_{p=0}^{l_{j,k}} (\alpha_{j,k,0}; \psi_{i_{j,k,1}}; \dots; \psi_{i_{j,k,p}}; \mathsf{pref}(\alpha_{j,k,p})) \vee \\ \bigvee_{p=1}^{l_{j,k}} (\alpha_{j,k,0}; \psi_{i_{j,k,1}}; \dots; \psi_{i_{j,k,p-1}}; \alpha_{j,k,p-1}; Y_{i_{j,k,p}}) \end{array} \right)$$

This can be established by a direct check with the definition of pref.

6.3 The Limits of the Expressibility of pref *and* suff

The valid equivalences about pref and suff from the previous two subsections entail that these operators can be expressed in the negation-free fragment of μHDC with the restriction on temporal propositional letters bound by a μ operator not to occur in the scope of a quantifier over individuals, nor in the scope of $(./.)$ in this μ operator's arguments.

Unfortunately, no general proof rule can be formulated about pref and suff. This is so, because $\models_{\mu HDC} \mathsf{pref}((\ell \neq 0; \varphi))$ is equivalent to the satisfiability of φ. Hence, only fragments of μHDC with pref with the same complexity of validity and satisfiability may have complete axiomatic systems. In particular, if validity is recursively enumerable and not recursive in some fragment of μHDC with negation, then satisfiability is not recursively enumerable due to the famous theorem of Post (cf. e.g. [Sho67]). Hence, such a fragment could not be recursively axoimatisable.

Remarks and Related Work

As we mentioned in the introduction, the first logic akin to DC to be extended by a projection operator was discrete-time ITL [HMM83,Mos86,Mos95]. Another interesting generalisation of an ITL projection operator introduced in [Mos86,Mos95] can be found in [He99,Gue00c].

The ideas behind projection onto state in DC can be traced back to an early variant of DC, where heterogenous time domains consisting of discrete computation *micro*time to specify the internal working of a controller, and dense *macro*time for the working of the controlled plant [PD97] were proposed. In that variant of DC there were two flexible constants ℓ and η to measure macro- and micro-time respectively. In our example of specification these constants can be defined as $\int \neg N$ and $\int N$ respectively. These duration terms equal ℓ in the scope of $(./\neg N)$ and $(./N)$ respectively.

Special cases of the prefix operator have been used earlier to abbreviate notation, see e.g. [Die96,DVH99]. The operators pref and suff can be regarded as non-deterministic versions of the pair of expanding modalities introduced to DC in [Pan96].

The semantics of pref, suff and projection onto state given here, and the proposed axioms and proof rules about these operators aim the greatest possible generality within real-time μHDC. On the contrary, the proposed way to specify concurrent temporal programs' behaviour has been tailored to employ as few of the extending features of DC as possible. In particular, nonterminating behaviour, which normally requires either expanding modalities or unbounded intervals to specify, has been dealt with by almost ordinary means - with the special condition on $[\![.]\!]'$ formulas to hold on all the bounded initial subintervals of the considered non-terminating behaviour only. The explicit account of computation time, which is compensated for by the possibility to use projection onto state for the formulation of requirements, has also enabled the specification of assignment without involving *super-dense chop* (cf. e.g. [HX99].)

A basic feature of μHDC which was not made use of in the example behaviour specification, but would certainly be needed to manage a fully-fledged programming language, is the state variable binding quantifier. It is needed to specify local variables (cf e.g. [HX99]) which are not included in our example language for the sake of simplicity. Local variables can commence in unlimited numbers due to recursive invocations, and therefore cannot be treated as some of the finitely many variables of global scope and extent which occur freely in formulas of the kind $[\![P]\!]_i$ and $[\![P]\!]'_i$. An appropriate clause of the definition of e.g. $[\![.]\!]_i$ for executing subprocess P with local variable p could be $[\![\mathtt{var}\ p; P]\!]_i \rightleftharpoons \exists p[\![P]\!]_i$

References

[AN01] Arnold, A and D. Niwiński *Rudiments of the μ-calculus.* Elsevier, 2001.

[Die00] Dierks, H. *Specification and Verification of Polling Real-Time Systems.* Ph.D. Thesis, Oldenburg University, 2000.

[Die96] Dietz, C. Graphical formalization of real-time requirements. *Proceedings of FTRTFT'96*, LNCS 1135, pp. 366-385, Springer-Verlag, 1996.

[DVH99] Dang Van Hung. *Projections: A Technique for Verifying Real-Time Programs in Duration Calculus.* Technical Report 178, UNU/IIST, P.O. Box 3058, Macau, November 1999.

[Gue00a] Guelev, D. P. A Complete Fragment of Higher-Order Duration μ-Calculus. *Proceedings of FST TCS 2000.* LNCS 1974, Sringer Verlag, 2000.

[Gue00b] Guelev, D. P. *Interpolation and related results on DC*.* Research Report 203, UNU/IIST, P.O.Box 3058, Macau, June 2000.

[Gue00c] Guelev, D. P. *A Complete Proof System for First Order ITL with Projection.* Research Report 202, UNU/IIST, P.O.Box 3058, Macau, June 2000.

[HMM83] Halpern, J., Z. Manna and B. Moszkowski. A Hardware Semantics Based on Temporal Intervals. *Proceedings of ICALP'83*, pp. 278-91, LNCS 154, Springer Verlag, 1983.

[He99] He Jifeng. A Behavioral Model for Co-design. *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems*, September, 1999, pp. 1420-1439, LNCS 1709, Springer Verlag, 1999.

[HX99] He Jifeng and Xu Qiwen. Advanced Features of DC and Their Applications. *Millenial Perspectives in Computer Science,* Palgrave, 2000.

[Mos86] Moszkowski, B. *Executing Temporal Logic Programs.* Cambridge University Press, 1986.

[Mos95] Moszkowski, B. Compositional Reasoning about Projected and Infinite Time. *Proceedings of ICECCS'95,* IEEE Computer Society Press, pp. 238-245, Los Alamitos, California, 1995.

[Pan95] Pandya, P. K. Some extensions to Mean-Value Calculus: Expressiveness and Decidability. *Proceedings of CSL'95*, LNCS 1092, 1995.

[Pan96] Pandya, P. K. Weak chop inverses and liveness in mean-value calculus. *Proceedings of FTRTFT'96*, LNCS 1135, 1996.

[PD97] Pandya, P. K. and Dang Van Hung. *Duration Calculus of Weakly Monotonic Time.* Technical Report 122, UNU/IIST, P.O.Box 3058, Macau, September 1997.

[Rav95] Ravn, A. P. *Design of Embedded Real-Time Computing Systems.* Technical report 1995-170, Technical University of Denmark, 1995.

[Sho67] Shoenfield, J. *Mathematical Logic.* Addison-Wesley, 1967.

[ZGZ99] Zhou Chaochen, D. P. Guelev and Zhan Naijun. A Higher-Order Duration Calculus. *Millenial Perspectives in Computer Science*, Palgrave, 2000.

[ZH96] Zhou Chaochen and M. Hansen Chopping a Point. *Proceedings of BCS FACS 7th Refinement Workshop, Electronic Workshop in Computer Sciences*, Springer-Verlag, 1996.

[ZHR91] Zhou Chaochen, C. A. R. Hoare and A. P. Ravn. A Calculus of Durations. *Information Processing Letters*, 40(5), pp. 269-276, 1991.

[ZHS93] Zhou Chaochen, M.R. Hansen and P. Sestoft. Decidability and Undecidability Results for Duration Calculus. *Proceedings of STACS'93*, Würzburg, LNCS 665, pp. 58-68, February 1993.

[ZRH93] Zhou Chaochen, A.P. Ravn and M.R. Hansen. An Extended Duration Calculus for Hybrid Real-Time Systems. *Hybrid Systems*, LNCS 736, Springer Verlag, 1993.

Remark: UNU/IIST reports are available through `http://www.iist.unu.edu`.