Int J Software Informatics, Vol.3, No.2-3, June/September 2009, pp. 171–196 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
©2009 by Institute of Software, Chinese Academy of Sciences. All rights reserved. Tel: +86-10-62661040

Efficient Model Checking for Duration Calculus*

Martin Fränzle[1] and Michael R. Hansen[2]

[1](Department of Computing Science, Carl von Ossietzky Universität Oldenburg, D-26111 Oldenburg, Germany)

[2](DTU Informatics, Technical University of Denmark, DK-2800 Lyngby, Denmark)

Abstract Duration Calculus (abbreviated to DC) is an interval-based, metric-time temporal logic designed for reasoning about embedded real-time systems at a high level of abstraction. But the complexity of model checking any decidable fragment featuring both negation and chop, DC's only modality, is non-elementary and thus impractical. Even worse, when such decidable fragments are generalized just slightly to cover more interesting durational constraints the resulting fragments become undecidable.

We here investigate a similar approximation as frequently employed in model checking situation- or point-based temporal logics, where linear-time problems are safely approximated by branching-time counterparts amenable to more efficient model-checking algorithms. Mimicking the role that a situation has in (A)CTL as the origin of a set of linear traces, we define a branching-time counterpart to interval-based temporal logics building on situation pairs spanning sets of intervals. While this branching-time interval semantics yields the desired reduction in complexity of the model-checking problem, from undecidable to linear in the size of the formula and cubic in the size of the model, the approximation is too coarse to be practical. We therefore refine the semantics by an occurrence count for crucial states (e.g., cuts of loops) in the model, arriving at a 4-fold exponential model-checking problem sufficiently accurately approximating the original one. Furthermore, when chop occurs in negative polarity only in DC formulas, we have a doubly exponential model-checking algorithm.

Key words: model checking; interval logics; duration calculus; branching time approximation

Fränzle M, Hansen MR. Efficient model checking for duration calculus. *Int J Software Informatics*, **2009, 3(2-3): 171–196. http://www.ijsi.org/1673-7288/3/171.htm**

1 Introduction

Duration Calculus (DC), as introduced by Zhou, Hoare, and Ravn[1] and thoroughly analyzed in Refs.[2, 3], is a metric-time temporal logic designed for reasoning about embedded real-time systems at a high level of abstraction, which primarily is achieved by basing the semantics on intervals rather than just temporal snapshots. While the resulting abstractness is desirable for specification and analysis, it is a

---

* This work has been partially funded by SFB/TR 14 AVACS (German Research Council), Velux Fonden, ARTIST2 (IST-004527), MoDES (Danish Research Council 2106-05-0022) and DaNES (the Danish National Advanced Technology Foundation).

Corresponding author: Martin Fränzle, Email: fraenzle@informatik.uni-oldenburg.de

Manuscript received 2009-02-28; revised 2009-08-05; accepted 2009-08-07.

burden for automatic verification support. Checking dense-time models against DC requires certain properties of the model, like the number of state changes being finitely bounded over finite intervals[4] or even finitely bounded along the whole run[5], unless the use of temporal operators or negation is seriously restricted[6−9]. Otherwise, the model property turns out to be undecidable[2,4].

Discrete-Time DC, i.e. DC interpreted over the natural numbers instead of $\mathbb{R}_{\geqslant 0}$, has more favorable decidability properties, as shown in Refs.[10, 11]. Here, decidability holds for various fragments featuring both negation and chop, provided that the use of durations is confined to simple cases where the accumulated duration of a state proposition is directly compared to a constant rather than building linear combinations of durations and relating these to constants, as permitted in more expressive fragments. There have been various attempts to build automatic verification support for discrete-time DC, e.g. Refs.[12-14]. But none of these systems has come to be routinely used for checking non-trivial formulas due to the extreme, non-elementary, complexity of deciding or model-checking DC formulas[2,4,15]. In Ref.[16] there is an interesting approach where QDDC, a discrete-time version of DC, is incorporated in CTL*. The result is a powerful logic capable of expressing liveness and branching properties as well as interval properties of the past. But model-checking remains non-elementary because the DC fragment is interpreted over linear traces.

In situation-based (or, synonymously, point-based) temporal logics, an approach towards enhancing model-checking techniques is to exploit the linear time vs. branching time dichotomy: While requirements are most naturally expressed in linear-time idioms, the actual verification is performed using safe branching-time approximations[17]. Such an approach yields reliable certificates while being considerably more efficient — linear-time rather than PSPACE in the size of the formula.

We shall investigate similar approximations for interval-based logics. Mimicking the role a situation has in (A)CTL as the origin of a set of linear traces, we define a branching-time counterpart to interval-based logics building on situation pairs spanning sets of intervals. This branching-time interval semantics yields the desired reduction in complexity of the model-checking problem, from non-elementary (when dealing with simple duration constraints only) or even undecidable (for linear combinations of durations) to linear in the size of the formula and $O(|V|^3)$ in the size of the model, but the approximation is too coarse to be really useful. We therefore refine the semantics to have finer control over traces. Adding an occurrence count for crucial states (e.g., cuts of loops) in the model, properties need no longer be true for all traces between the situation pair under investigation, but there is a filter mechanism permitting just to focus on traces with certain properties. This filtering approach is a compromise between the prima-facie trace underlying the linear-time interpretation and the homogeneous treatment of complete trace ensembles underlying CTL-like branching-time interpretations. This approximation of linear-time is for many practical purposes sufficiently accurate. We provide a model-checking algorithm with a 4-fold exponential upper bound, as opposed to the non-elementary model-checking problem of linear-time DC. Furthermore, for DC-formulas having chop in negative polarity only, we give a doubly exponential model-checking algorithm.

This paper is an extended version of Ref.[18]. (1) The fragment of DC considered here is significantly more expressive, addressing a fragment which is undecidable already in the discrete-time setting[10,3]. (2) The filtering mechanism for the traces has been refined thereby supporting finer approximations. (3) The model-checking algorithm is significantly improved using a solver for integer LinSAT, i.e. Boolean combinations of linear arithmetic constraints over the integers[19], for an interesting class of formulas.

2 Duration Calculus

Duration Calculus (DC) is a logic specifically tailored for reasoning about embedded real-time systems at a high level of abstraction from operational detail. We give a brief introduction to the discrete-time DC fragment used in this paper. For more complete coverage of the topic we refer to Refs.[3, 20, 21].

2.1 Syntax

The syntax is defined below with two syntax categories: *state expressions*, ranged over by $S, S_1, S_2, \ldots$, and *formulas*, ranged over by $\phi, \phi_1, \psi, \psi_1, \ldots$ State expressions are Boolean combinations of *state variables*, and they describe combined states of a system at a given point in time. Formulae can be considered as truth-valued functions on time intervals. The syntax is defined as follows:

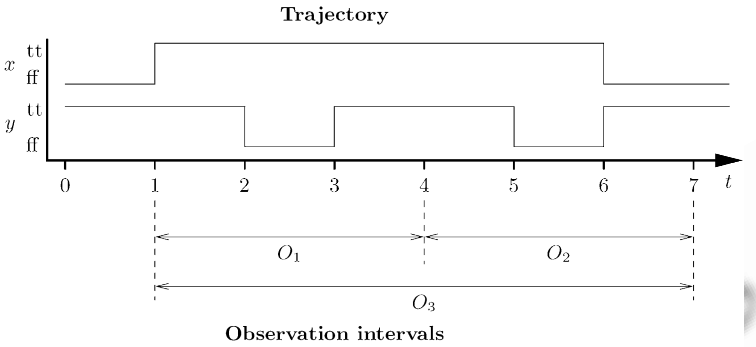
$$\begin{array}{rcl} S & ::= & 0 \mid 1 \mid P \mid \neg S \mid S_1 \vee S_2\ , \\ \phi & ::= & \top \mid \Sigma_{i\in\Omega} c_i \textstyle\int S_i \bowtie k \mid \neg\phi \mid \phi \wedge \psi \mid \phi \frown \psi\ , \end{array}$$

where $P \in \mathit{StateVar}$, with $\mathit{StateVar}$ being a countable set of state variable names, $k, c_i \in \mathbb{Z}$, $\bowtie \in \{<, \leqslant, =, \geqslant, >\}$, and $\Omega \subset_{fin} \mathbb{N}$.

While the meaning of the Boolean connectives used in DC formulas should be obvious, the temporal connective $\frown$ (pronounced "chop"), which is inherited from Interval Temporal Logic [22], may need some explanation. Formulae are interpreted over, first, trajectories providing valuation of state variables that varies over time and, second, over finite intervals of time, called "observation intervals". A formula $\phi \frown \psi$ is true of an observation interval iff the observation interval can be split into a left and a right subinterval s.t. ϕ holds of the left part and ψ of the right part. A duration formula $2\int P + -1\int Q > 3$ is true of an observation interval iff the twice the accumulated duration of state assertion P, interpreted over the trajectory, being true over the observation interval minus the accumulated duration of state assertion Q being true over the observation interval exceeds 3. Fig. 1 provides an illustration of the meaning of these formulas.

2.2 Semantics

An interpretation $\mathcal{I}$ associates a function $P_{\mathcal{I}} : \mathbb{N} \to \{0, 1\}$ with every state variable P, where $\mathbb{N}$ models the discrete time line. The relationship between an interpretation for a state variable, say P, and a trajectory, as shown in Fig. 1, is that $P_{\mathcal{I}}(t) = 1, t \in \mathbb{N}$, iff the trajectory for P is high throughout the real interval $[t, t + 1)$. The semantics of a state expression S, given an interpretation $\mathcal{I}$, is a function: $\mathcal{I}[\![S]\!] : \mathbb{N} \to \{0, 1\}$:



The formula $(1\int x \geqslant 3) \wedge (1\int y < 3)$ holds on observation interval $O_1 = [1,4]$, as the accumulated duration of x being true over this interval is 3 and that of y being true over this interval is 2. Analogously, $1\int y + -2\int(x \wedge \neg y) = 0$ holds on observation interval $O_2 = [4,7]$. Therefore, the formula $((1\int x \geqslant 3) \wedge (1\int y < 3)) \frown (1\int y + -2\int(x \wedge \neg y) = 0)$ holds on the catenation $O_3 = [1,7]$ of the other two observation intervals.

Figure 1. The meaning of accumulated durations and of the chop modality

$$
\begin{aligned}
\mathcal{I}[\![0]\!](t) &= 0\\
\mathcal{I}[\![1]\!](t) &= 1\\
\mathcal{I}[\![P]\!](t) &= P_{\mathcal{I}}(t)\\
\mathcal{I}[\![\neg S]\!](t) &= \begin{cases} 0 & \text{if } \mathcal{I}[\![S]\!](t) = 1\\ 1 & \text{if } \mathcal{I}[\![S]\!](t) = 0 \end{cases}\\
\mathcal{I}[\![S_1 \vee S_2]\!](t) &= \begin{cases} 0 & \text{if } \mathcal{I}[\![S_1]\!](t) = \mathcal{I}[\![S_2]\!](t) = 0\\ 1 & \text{otherwise.} \end{cases}
\end{aligned}
$$

Satisfaction of formulas ϕ is defined over pairs $(\mathcal{I}, [a,b])$ (called *observations*) of an interpretation $\mathcal{I}$ and a time interval $[a,b]$ with $a \leqslant b$ and $a, b \in \mathbb{N}$. The satisfaction relation $\mathcal{I}, [a,b] \models \phi$ is defined as follows:

$$
\begin{array}{lll}
\mathcal{I}, [a,b] \models \top & & \\
\mathcal{I}, [a,b] \models \Sigma_{i\in\Omega} c_i \int S_i \bowtie k & \text{iff} & \Sigma_{i\in\Omega}\left(c_i \sum_{t=a}^{b-1} \mathcal{I}[\![S_i]\!](t)\right) \bowtie k\\
\mathcal{I}, [a,b] \models \neg\phi & \text{iff} & \mathcal{I}, [a,b] \not\models \phi\\
\mathcal{I}, [a,b] \models \phi \wedge \psi & \text{iff} & \mathcal{I}, [a,b] \models \phi \text{ and } \mathcal{I}, [a,b] \models \psi\\
\mathcal{I}, [a,b] \models \phi \frown \psi & \text{iff} & \mathcal{I}, [a,m] \models \phi \text{ and } \mathcal{I}, [m,b] \models \psi\\
 & & \text{for some } m \in [a,b] \cap \mathbb{N}.
\end{array}
$$

Whenever $\mathcal{I}, [a,b] \models \phi$ holds we say that ϕ is *true* in $[a,b]$ wrt. $\mathcal{I}$. A formula ϕ is *valid* (written $\models \phi$) if $\mathcal{I}, [a,b] \models \phi$ holds for every observation $(\mathcal{I}, [a,b])$, and ϕ is *satisfiable* if $\mathcal{I}, [a,b] \models \phi$ for some observation $(\mathcal{I}, [a,b])$.

Based on the temporal connective $\frown$ (called "chop") inherited from Interval Temporal Logic[22], the temporal operators $\Diamond$ and $\Box$, meaning 'in some subinterval' and 'in every subinterval', can be defined by: $\Diamond\phi \overset{\text{df}}{=} ((\top \frown \phi) \frown \top)$ and $\Box\phi \overset{\text{df}}{=} \neg\Diamond\neg\phi$.

The following formulas give a flavor of the kind of durational properties which can be expressed using the primitives of DC.

ℓ	$\overset{\text{df}}{=}$	$\int 1$	‘interval length’
$1\int S \geqslant k$			‘duration of S is at least k in an interval’
$\ell = k$	$\overset{\text{df}}{=}$	$1\int 1 = k$	‘interval length is k’
$\int S_1 = \int S_2$	$\overset{\text{df}}{=}$	$1\int S_1 + -1\int S_2 = 0$	‘S_1 and S_2 have the same duration in an interval’
$\llbracket S \rrbracket$	$\overset{\text{df}}{=}$	$1\int \neg S = 0 \wedge \ell > 0$	‘S holds through a non-point interval’

These constructions and the formulas that can be built using them demonstrate that the DC fragment we consider is quite expressive, despite its simple syntax. As an example take the formula $\Box(\ell \geqslant 10 \Rightarrow (\int p - 2\int q \leqslant 1 \wedge \int p - 2\int q \geqslant -1)^1$, which expresses that over all sufficiently long time frames ($\ell \geqslant 10$), process P is granted roughly twice the run-time of process Q, if observables p and q are considered to represent the time instances where the respective processes are executed.

It has been shown in Ref.[10] that the satisfiability problem for the above fragment of DC is undecidable already over discrete time, as the linear combinations of durations permit to encode unbounded counters. Without weighted sums of durations, i.e. with atomic formulas of the form $\int S \bowtie k$ only, the discrete-time satisfiability problem is decidable with non-elementary complexity due to a reduction to the emptiness problem of extended regular expressions[10]. To this end, note that chop coincides to concatenation of languages, $\neg$ to complement, $\wedge$ to intersection, and that the traces satisfying an atomic formula of the form $\int S \bowtie k$ can be defined by a regular expression. It has furthermore been shown that satisfiability by a dense-time trace of finitely bounded variability (in the sense of having a fixed, uniform bound k on the number of state changes in any unit interval of time) is also decidable for this fragment[4].

2.3 Labelled Kripke structures

We shall consider the satisfaction problem with respect to interpretations which are generated by Kripke structures.

Definition 1. *A labelled Kripke structure* (*or just Kripke structure*) *is a tuple* $K = (V, E, I, L)$*, where*

- V *is a finite set whose elements are called vertices or situations,*
- $E \subseteq V \times V$ *is a set of edges or transitions,*
- $I \subseteq V$ *is a non-empty set of initial vertices* (*or situations*)*, and*
- $L : V \rightarrow 2^{StateVar}$ *is a labelling function assigning to each vertex a set of state variables* (*of DC*)*. A vertex* v *fulfills state expression* S*, written* $v \models_L S$*, iff* S *is satisfied when the state variables in* $L(v)$ *are assigned the value* 1 *and those not in* $L(v)$ *the value* 0.

We shall consider the following kinds of behaviour for Kripke structures:

---

[1] Here and in the sequel, we drop factors of 1 in linear combinations and merge negative factors with the preceding additive operator.

Definition 2. *Let $K = (V, E, I, L)$ be a labelled Kripke structure.*

- *A trace (or run) tr of K from i to j ($i, j \in V$) is a non-empty, finite sequence of vertices: $v_0\, v_1 \ldots v_k$ where $v_0 = i, v_k = j$, and $(v_l, v_{l+1}) \in E$, for $0 \leqslant l < k$. Let $\operatorname{img} tr$ be the set of vertices $\{v_0, v_1, \ldots, v_k\}$.*
- *The interpretation induced by tr, denoted $\mathcal{I}_{tr}$, is defined for any $t \in \mathbb{N}$ by*

$$P_{\mathcal{I}_{tr}}(t) = \begin{cases} 1 & \text{if } t \leqslant k \text{ and } P \in L(v_t), \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3. *Given a labelled Kripke structure $K = (V, E, I, L)$, a DC formula ϕ, and a trace $tr = v_0\, v_2 \cdots v_k$ of K.*

- *The trace tr satisfies ϕ, written $tr \models_K \phi$, if $\mathcal{I}_{tr}, [0, k] \models \phi$.*
- *The Kripke structure K is a model of ϕ, written $K \models \phi$, if for every trace tr of K starting from a vertex in I we have that $tr \models_K \phi$.*

The model-checking problem $K \models \phi$ is undecidable. This follows directly from the undecidability of the satisfiability problem of the considered fragment of DC, since we can define Kripke structures which can generate all interpretations for the state variables occurring in a formula on arbitrary intervals. The aim of this paper is to develop a safe approximation technique for the model-checking problem in order to achieve efficient tool support.

3 Doubly Situation-Based Semantics

Given the complexity gap between model checking linear-time temporal logics and situation-based, branching-time logics, it seems attractive to define a situation-based variant of DC that can be used in the same way ACTL is frequently used as an efficiently analyzable substitute for PLTL (cf., e.g., Ref.[17]). We provide such a situation-based interpretation of DC. As DC is a logic of intervals rather than temporal snapshots, the natural counterpart to CTL's notion of a situation i, which gives rise to a computation tree rooted in i, is a pair of situations (i, j) giving rise to a bundle of runs originating in i and ending in j. This ensemble of runs corresponds to a set of observations obtained using the labelling function of the Kripke structure.

The *doubly situation-based semantics* is a function $K[\![\phi]\!] : V \to V \to 2^{\mathbb{B}}$ mapping a pair of situations to the power domain of truth values $2^{\mathbb{B}}$, where the singleton sets are interpreted as definite truth values while the other subsets of $\mathbb{B}$ denote the following situations: value $\emptyset$ represents the fact that there are some traces between the situation pair of interest satisfying and others violating the formula such that no meaningful truth value can be assigned wrt. a universally path-quantified property. $\mathbb{B}$, on the contrary, denotes the scenario that all traces between the situations do satisfy *and* violate the formula, which can only happen if there is no connecting trace. The semantic definition is then given by:

$$K[\![\top]\!]\, i\, j = \begin{cases} \{\texttt{true}\} & \text{if } j \text{ is reachable from } i \\ \mathbb{B} & \text{otherwise} \end{cases}$$

$$K\llbracket \Sigma_{i\in\Omega} c_i \textstyle\int S_i \bowtie k \rrbracket\, i\, j \;=\; \begin{cases} \mathbb{B} & \text{if } j \text{ is unreachable from } i \\ \{\mathtt{true}\} & \text{if } tr \models_K \Sigma_{i\in\Omega} c_i \int S_i \bowtie k \\ & \quad \text{for all runs } tr \text{ of } K \text{ from } i \text{ to } j \\ \{\mathtt{false}\} & \text{if } tr \not\models \Sigma_{i\in\Omega} c_i \int S_i \bowtie k \\ & \quad \text{for all runs } tr \text{ of } K \text{ from } i \text{ to } j \\ \emptyset & \text{otherwise} \end{cases}$$

$$K\llbracket \neg\phi \rrbracket\, i\, j \;=\; \begin{cases} \{\mathtt{true}\} & \text{if } K\llbracket\phi\rrbracket\, i\, j = \{\mathtt{false}\}, \\ \{\mathtt{false}\} & \text{if } K\llbracket\phi\rrbracket\, i\, j = \{\mathtt{true}\} \\ K\llbracket\phi\rrbracket\, i\, j & \text{otherwise} \end{cases}$$

$$K\llbracket \phi \wedge \psi \rrbracket\, i\, j \;=\; \begin{cases} \{\mathtt{true}\} & \text{if } K\llbracket\phi\rrbracket\, i\, j = K\llbracket\psi\rrbracket\, i\, j = \{\mathtt{true}\} \\ \{\mathtt{false}\} & \text{if } K\llbracket\phi\rrbracket\, i\, j \text{ or } K\llbracket\psi\rrbracket\, i\, j \text{ is } \{\mathtt{false}\} \\ K\llbracket\phi\rrbracket\, i\, j \cap K\llbracket\psi\rrbracket\, i\, j & \text{otherwise} \end{cases}$$

$$K\llbracket \phi \frown \psi \rrbracket\, i\, j \;=\; \begin{cases} \mathbb{B} & \text{if } j \text{ is unreachable from } i \\ \{\mathtt{true}\} & \text{if for each run } tr \text{ of } K \text{ from } i \text{ to } j \\ & \text{there is } k \in \operatorname{img} tr \text{ with:} \\ & K\llbracket\phi\rrbracket\, i\, k = K\llbracket\psi\rrbracket\, k\, j = \{\mathtt{true}\} \\ \{\mathtt{false}\} & \text{if for each run } tr \text{ of } K \text{ from } i \text{ to } j \\ & \text{and for each } k \in \operatorname{img} tr \text{ we have :} \\ & K\llbracket\phi\rrbracket\, i\, k = \{\mathtt{false}\} \text{ or } K\llbracket\psi\rrbracket\, k\, j = \{\mathtt{false}\} \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that for a pair (i, j) of situations for which j is not reachable from i, $K\llbracket\phi\rrbracket\, i\, j = \{\mathtt{true}, \mathtt{false}\}$, representing the fact that all traces vacuously do both satisfy and violate the property. Otherwise, there is at least one run connecting i with j. The semantics then reports $\{\mathtt{true}\}$ if every run from i to j satisfies ϕ, it reports $\{\mathtt{false}\}$ if every run from i to j falsifies ϕ. In the remaining case where some runs satisfy ϕ while others falsify ϕ the semantics reports $\emptyset$.

Notice that negation and conjunction are interpreted by the strongest monotonic extensions to $2^{\mathbb{B}}$ of the respective Boolean connectives. Furthermore, $K\llbracket\phi \frown \psi\rrbracket\, i\, j$ is $\{\mathtt{true}\}$ iff on each run from i to j there is an intermediate situation m such that (i, m) and (m, j) yield $\{\mathtt{true}\}$ for ϕ and ψ, respectively. Similarly, it is $\{\mathtt{false}\}$ iff on any such run from i to j, any intermediate situation yields $\{\mathtt{false}\}$ for at least one of ϕ or ψ.

Lemma 1 (Approximation). Let $K = (V, E, I, L)$ be a Kripke structure, $i, j \in V$ two situations, and tr a run of K from i to j. Then

1. $K\llbracket\phi\rrbracket\, i\, j = \{\mathtt{true}\}$ implies $tr \models_K \phi$,
2. $K\llbracket\phi\rrbracket\, i\, j = \{\mathtt{false}\}$ implies $tr \not\models_K \phi$,

i.e., if the situation-based semantics provides a definite truth value then this truth value coincides with that assigned by the linear-time semantics to *all* runs between the respective situations.

Proof. Straightforward induction on the structure of ϕ. $\square$

For Kripke structures where there is at most one run between any pair of situations, the doubly situation-based semantics is precise, and we get the following result.

Lemma 2 (Preciseness) . Let $K = (V, E, I, L)$ be a Kripke structure, where E constitutes a set of trees, $i, j \in V$ are two situations, and tr is a run of K from i to j. Then

1. $tr \models_K \phi$ implies $K[\![\phi]\!]\, i\, j = \{\texttt{true}\}$,
2. $tr \not\models_K \phi$ implies $K[\![\phi]\!]\, i\, j = \{\texttt{false}\}$.

3.1 Model checking

We present an algorithm for checking whether a Kripke structure $K = (V, E, I, L)$ satisfies a formula ϕ wrt. the doubly situation-based semantics. The time-complexity of this algorithm is $O(|V|^3)$ in the size of the Kripke structure K and linear in the size of ϕ. The algorithm proceeds by induction on the structure of ϕ, marking situation pairs (i, j) with the subformulas of ϕ and the associated values. I.e., marking a situation pair (i, j) with *just* $(\psi, b), b \in \mathbb{B}$ means that $K[\![\psi]\!]\, i\, j = \{b\}$, and a mark $(\psi, \emptyset)$ is provided whenever $K[\![\psi]\!]\, i\, j = \emptyset$.

In order to deal with the case where j is not reachable from i, i.e. $K[\![\phi]\!]\, i\, j = \{\texttt{true}, \texttt{false}\}$, we mark such situation pairs (i, j) with both $(\phi, \texttt{true})$ and $(\phi, \texttt{false})$. This is done in an initialization step and the time complexity of that is $O(|V|^3)$. When we consider a situation pair (i, j) below, we tacitly assume that j is reachable from i.

1. *Base case* $\phi = \top$. As $\top$ is satisfied by all pairs (i, j), all such pairs become marked with $(\top, \texttt{true})$. These markings can be performed in $O(|V|^2)$ time.
2. *Base case* $\phi = \Sigma_{i\in\Omega} c_i \textstyle\int S_i < k$. The other cases of $\Sigma_{i\in\Omega} c_i \int S_i \bowtie k$ are similar. Let $\Delta(v)$ denote the contribution of $\Sigma_{i\in\Omega} c_i \int S_i$ when staying one time unit in the vertex v, i.e. $\Delta(v) = \Sigma_{j\in\{i\in\Omega | v\models_L S_i\}} c_j$. Associate to K a weighted graph which has vertices V and edge weights

$$w(v_1, v_2) = \begin{cases} \infty & \text{iff } (v_1, v_2) \notin E, \\ W(v_1) & \text{iff } (v_1, v_2) \in E, \end{cases}$$

   where $W(v_1) = \Delta(v_1)$ if we are determining lower bounds on $\Sigma_{i\in\Omega} c_i \int S_i$ (or $W(v_1) = -\Delta(v_1)$, respectively, for upper bounds). We can then use an all-pairs shortest-path algorithm to determine the lower bound (upper bound, resp.) of $\Sigma_{i\in\Omega} c_i \int S_i$ along all paths between all pairs of vertices in K. This part takes $O(|V|^{2.575})$ time using Zwick's algorithm for directed graphs with integer edge weights of small absolute value[23]. We assign markings to all pairs (i, j) within one sweep over $V \times V$ as follows:

   - mark (i, j) with $(\Sigma_{i\in\Omega} c_i \int S_i < k, \texttt{true})$ iff the *upper* bound on $\Sigma_{i\in\Omega} c_i \int S_i$ along paths between i and j is strictly less than k,
   - mark (i, j) with $(\Sigma_{i\in\Omega} c_i \int S_i < k, \texttt{false})$ iff the *lower* bound on $\Sigma_{i\in\Omega} c_i \int S_i$ on paths from i to j is greater than or equal to k,
   - otherwise, mark (i, j) with $(\Sigma_{i\in\Omega} c_i \int S_i < k, \emptyset)$.

The complexity of the second step is $O(|V|^2)$ such that the overall complexity coincides with that of the all-pairs shortest-path algorithm.

3. *Case* $\phi = \neg\psi$. We assign values for $\neg\psi$ to all pairs (i, j) within one sweep over $V \times V$ taking $O(|V|^2)$ time as follows:
   - mark (i, j) with $(\neg\psi, \texttt{true})$ iff (i, j) is marked with $(\psi, \texttt{false})$,
   - mark (i, j) with $(\neg\psi, \texttt{false})$ iff (i, j) is marked with $(\psi, \texttt{true})$,
   - otherwise, we mark (i, j) with $(\neg\psi, \emptyset)$.
4. *Case* $\phi = \chi \wedge \psi$. We assign values for $\chi \wedge \psi$ to all pairs (i, j) within one sweep over $V \times V$ taking $O(|V|^2)$ time as follows:
   - mark (i, j) with $(\chi \wedge \psi, \texttt{true})$ iff both $(\chi, \texttt{true})$ and $(\psi, \texttt{true})$ are markings of (i, j),
   - mark (i, j) with $(\chi \wedge \psi, \texttt{false})$ iff $(\chi, \texttt{false})$ is a marking of (i, j) or $(\psi, \texttt{false})$ is a marking of (i, j),
   - otherwise, mark (i, j) with $(\chi \wedge \psi, \emptyset)$.
5. *Case* $\phi = \chi \frown \psi$. We assign values for $\chi \frown \psi$ to all pairs (i, j) as follows:
   - mark (i, j) with $(\chi \frown \psi, \texttt{true})$ iff i and j are unconnected in the graph (V, E') obtained from (V, E) by removing all situations $k \in V$ which have both (i, k) marked with $(\chi, \texttt{true})$ and (k, j) with $(\psi, \texttt{true})$,
   - mark (i, j) with $(\chi \frown \psi, \texttt{false})$ iff for all $k \in V$ reachable from i and backward reachable from j, (i, k) is marked with $(\chi, \texttt{false})$ or (k, j) with $(\psi, \texttt{false})$,
   - otherwise, we mark (i, j) with $(\chi \frown \psi, \emptyset)$.

   The complexity of this step is $O(|V|^3)$.

As the algorithm recurs over all subformulas of ϕ in order to determine the situation pairs satisfying ϕ, the overall complexity of this algorithm is $O(|\phi||V|^3)$.

Theorem 1 (Correctness of model checking algorithm). Let $K = (V, E, I, L)$ be a Kripke structure, $i, j \in V$ two situations, and ϕ a DC formula.

If $K[\![\phi]\!]\,i\,j = \{x\}, x \in \mathbb{B}$ then the model-checking algorithm marks (i, j) with (ϕ, x) and there is no dissenting marking (ϕ, y) of (i, j) with $x \neq y$.

If $K[\![\phi]\!]\,i\,j = \emptyset$ then the model-checking algorithm marks (i, j) with $(\phi, \emptyset)$ and there is no other marking of (i, j) for ϕ.

If j is not reachable from i in K and, hence, $K[\![\phi]\!]\,i\,j = \{\texttt{true}, \texttt{false}\}$, then the algorithm marks (i, j) with both $(\phi, \texttt{true})$ and $(\phi, \texttt{false})$.

Proof. The proof is by induction on the syntactic structure of ϕ. Correctness of the base case $\phi = \top$ is obvious, and so is that the all-pairs shortest-path algorithm reports the correct bounds on $\Sigma_{i \in \Omega} c_i \int S_i$. Furthermore, the two recursive cases $\phi = \neg\psi$ and $\phi = \chi \wedge \psi$ are simple.

For the correctness of the algorithm for chop, observe that $K[\![\chi \frown \psi]\!]\,i\,j = \{\texttt{true}\}$ iff *all* runs from i to j visit some run-dependent chop point m with $K[\![\chi]\!]\,i\,m = K[\![\psi]\!]\,m\,j = \{\texttt{true}\}$. Taking the graph (V, E') obtained from (V, E) by removing all vertices $m \in V$ which have (i, m) marked with $(\chi, \texttt{true})$ and (m, j) marked with

$(\psi, \texttt{true})$, we have that j is reachable from i in (V, E') iff there is some run from i to j not passing through any state k with (i, k) marked with $(\chi, \texttt{true})$ and (k, j) marked with $(\psi, \texttt{true})$. By induction hypothesis, this implies that j is reachable from i in (V, E') iff there is some run tr from i to j s.t. all situations k visited by tr have $K\llbracket\chi\rrbracket\, i\, k \neq \{\texttt{true}\}$ or $K\llbracket\psi\rrbracket\, k\, j \neq \{\texttt{true}\}$, i.e. iff $K\llbracket\phi\rrbracket\, i\, j \neq \{\texttt{true}\}$. Consequently, the algorithm marks (i, j) with $(\chi \frown \psi, \texttt{true})$ iff $K\llbracket\chi \frown \psi\rrbracket = \{\texttt{true}\}$ or j is not reachable from i in K.

The remaining parts are easily proved. □

Example: Consider the Kripke structure in Fig. 2, which we want to model check against the formula $\phi = \neg(\top \frown (\int 1 < 4 \land \neg \int p < 3) \frown \top)$, which is the unfolding of the abbreviations in $\Box(\ell < 4 \Rightarrow \int p < 3)$.

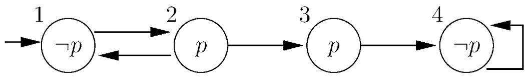


Figure 2. A sample Kripke structure

The markings produced by the model-checking algorithm are shown in Table 1.

Table 1 Markings for the Kripke structure from Fig. 2.

i, j	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	$(\top, \texttt{true})$ $(\#, \emptyset)$	$(\top, \texttt{true})$ $(\#, \emptyset)$	$(\top, \texttt{true})$ $(\#, \emptyset)$	$(\top, \texttt{true})$ $(\#, \emptyset)$
$i = 2$	$(\top, \texttt{true})$ $(\#, \emptyset)$	$(\top, \texttt{true})$ $(\#, \emptyset)$	$(\top, \texttt{true})$ $(\#, \emptyset)$	$(\top, \texttt{true})$ $(\#, \emptyset)$
$i = 3$	$(*, *)$	$(*, *)$	$(\top, \texttt{true})$ $(\int 1 < 4, \texttt{true})$ $(\int p < 3, \texttt{true})$ $(\neg \int p < 3, \texttt{false})$ $(\xi, \texttt{false})$ $(\top \frown \xi, \texttt{false})$ $((\top \frown \xi) \frown \top, \texttt{false})$ $(\phi, \texttt{true})$	$(\top, \texttt{true})$ $(\int 1 < 4, \emptyset)$ $(\int p < 3, \texttt{true})$ $(\neg \int p < 3, \texttt{false})$ $(\xi, \texttt{false})$ $(\top \frown \xi, \texttt{false})$ $((\top \frown \xi) \frown \top, \texttt{false})$ $(\phi, \texttt{true})$
$i = 4$	$(*, *)$	$(*, *)$	$(*, *)$	$(\top, \texttt{true})$ $(\int 1 < 4, \emptyset)$ $(\int p < 3, \texttt{true})$ $(\neg \int p < 3, \texttt{false})$ $(\xi, \texttt{false})$ $(\top \frown \xi, \texttt{false})$ $((\top \frown \xi) \frown \top, \texttt{false})$ $(\phi, \texttt{true})$

In this table, $(*, *)$ denotes (ψ, x) for each subformula ψ of ϕ and each $x \in \{\texttt{true}, \texttt{false}\}$, and $\#$ stands for all subformulas different from the aforementioned subformulas. Furthermore, $\xi = (\int 1 < 4 \land \neg \int p < 3)$ and $\phi = \neg((\top \frown \xi) \frown \top)$. Unfortunately, the approximation by the doubly situation-based semantics is too coarse to decide ϕ, as the state pairs $(1, 1)$ to $(1, 4)$, which represent the traces anchored in the initial vertex 1, all become marked with $(\phi, \emptyset)$.

4 Counting Semantics

To reduce the information loss encountered in the situation-based approximation, we will add information on the frequency of visits to certain crucial vertices in the Kripke structure, e.g. to cut points of loops such that the number of iterations of loops can be determined. Let $K = (V, E, I, L)$ be a Kripke structure.

We define a *multiset* m on V as a partial function

$$m : V \xrightarrow{\text{part}} \mathbb{N}$$

Let Mset denote the set of all multisets on V.

The main idea is that m conveys information about how many times a given vertex is visited. For example, if $m(i)$ is defined ($i \in \operatorname{dom} m$) and equal to n, then vertex i is visited n times in the concerned traces. If $m(i)$ is undefined ($i \notin \operatorname{dom} m$) then we do not know how many times i is visited. For a run $v_0\, v_1\, \ldots\, v_k$, the number of times vertex v *is visited* is $n = |\{i \in \mathbb{N} \mid i < k \wedge v = v_i\}|$, i.e. we disregard the last vertex, as that vertex does not contribute to the duration of any state variable for that run.

A run in K from i to j is *consistent* with m if, for every $k \in \operatorname{dom} m$, k is visited $m(k)$ times in the run. The set of m-consistent runs from i to j is denoted $R_{i,j,m}$ in the sequel. With $V_{i,j,m}$ we denote the set of all vertices visited on some m-consistent runs from i to j in K, i.e. $V_{i,j,m} = \bigcup_{tr \in R_{i,j,m}} \operatorname{img} tr$. Furthermore, we call K *consistent wrt.* i, j *and* m if there is a run from i to j which is consistent with m, i.e. if $R_{i,j,m} \neq \emptyset$.

A *consistent decomposition* of m wrt. K and three situations i, j, k is a pair of multisets m_1, m_2 where $\operatorname{dom} m = \operatorname{dom} m_1 = \operatorname{dom} m_2$ and $m = m_1 + m_2$ and K is consistent wrt. i, k, m_1 and wrt. k, j, m_2.

The *counting semantics* is a function $K[\![\phi]\!]_c : V \to V \to \text{Mset} \to 2^{\mathbb{B}}$:

$$K[\![\top]\!]_c\, i\, j\, m = \begin{cases} \mathbb{B} & \text{if } R_{i,j,m} = \emptyset \\ \{\texttt{true}\} & \text{if } R_{i,j,m} \neq \emptyset \end{cases}$$

$$K[\![\Sigma_{i\in\Omega} c_i \textstyle\int S_i \bowtie k]\!]_c\, i\, j\, m = \begin{cases} \mathbb{B} & \text{if } R_{i,j,m} = \emptyset \\ \{\texttt{true}\} & \text{if } R_{i,j,m} \neq \emptyset \text{ and} \\ & \text{for all } tr \in R_{i,j,m} : tr \models_K \Sigma_{i\in\Omega} c_i \int S_i \bowtie k \\ \{\texttt{false}\} & \text{if } R_{i,j,m} \neq \emptyset \text{ and} \\ & \text{for all } tr \in R_{i,j,m} : tr \not\models_K \Sigma_{i\in\Omega} c_i \int S_i \bowtie k \\ \emptyset & \text{otherwise} \end{cases}$$

$$K[\![\neg\phi]\!]_c\, i\, j\, m = \begin{cases} \{\texttt{true}\} & \text{if } K[\![\phi]\!]_c\, i\, j\, m = \{\texttt{false}\} \\ \{\texttt{false}\} & \text{if } K[\![\phi]\!]_c\, i\, j\, m = \{\texttt{true}\} \\ K[\![\phi]\!]_c\, i\, j\, m & \text{otherwise} \end{cases}$$

$$K[\![\phi \wedge \psi]\!]_c\, i\, j\, m = \begin{cases} \mathbb{B} & \text{if } R_{i,j,m} = \emptyset \\ \{\texttt{true}\} & \text{if } K[\![\phi]\!]_c\, i\, j\, m = K[\![\psi]\!]_c\, i\, j\, m = \{\texttt{true}\} \\ \{\texttt{false}\} & \text{if } K[\![\phi]\!]_c\, i\, j\, m \text{ or } K[\![\psi]\!]_c\, i\, j\, m \text{ is } \{\texttt{false}\} \\ \emptyset & \text{otherwise} \end{cases}$$

$$K\llbracket \phi \frown \psi \rrbracket_c \, i\, j\, m = \begin{cases} \mathbb{B} & \text{if } R_{i,j,m} = \emptyset \\ \{\texttt{true}\} & \text{if there is a } k \in V_{i,j,m} \text{ such that} \\ & \text{for each consistent decomposition} \\ & m_1, m_2 \text{ of } m \text{ wrt. } i, j, k: \\ & K\llbracket \phi \rrbracket_c \, i\, k\, m_1 = K\llbracket \psi \rrbracket_c \, k\, j\, m_2 = \{\texttt{true}\} \\ \{\texttt{false}\} & \text{if for each } k \in V_{i,j,m} \text{ and for each} \\ & \text{consistent decomposition } m_1, m_2 \text{ of } m \\ & \text{wrt. } i, j, k: \\ & K\llbracket \phi \rrbracket_c \; i\, k\, m_1 = \{\texttt{false}\} \text{ or} \\ & K\llbracket \psi \rrbracket_c \, k\, j\, m_2 = \{\texttt{false}\} \\ \emptyset & \text{otherwise} \end{cases}$$

Note that while a witness vertex is existentially chosen in the semantics of chop, the decompositions of the multisets are universally quantified. This is for the sake of consistency as we are only dealing with universal path properties such that each possible, i.e. compatible with the counts in the multiset, path through the witness vertex has to be inspected.

This semantic function extends the previous semantic function $K\llbracket \phi \rrbracket$ by considering m-consistent traces only. For example, $K\llbracket \phi \rrbracket_c \, i\, j\, m = \{\texttt{true}\}$ when every m-consistent trace from i to j in K satisfies ϕ.

Lemma 3 (Approximation). Let $K = (V, E, I, L)$ be a Kripke structure, $i, j \in V$, and $m \in \text{Mset}$. Then

1. $K\llbracket \phi \rrbracket_c \, i\, j\, m \supseteq \{\texttt{true}\}$ implies $\forall tr \in R_{i,j,m} : tr \models_K \phi$,
2. $K\llbracket \phi \rrbracket_c \, i\, j\, m \supseteq \{\texttt{false}\}$ implies $\forall tr \in R_{i,j,m} : tr \not\models_K \phi$,

i.e., if the counting semantics provides a definite truth value in the form of a singleton set then this truth value coincides with that assigned by the linear-time semantics to *all* runs consistent with the respective situations and multiset.

Proof. Straightforward induction on the structure of ϕ. □

For an acyclic Kripke structure and a fully defined multiset m with dom $m = V$, there is at most one m-consistent run between any pair of situations, and the counting semantics is in this case precise.

Lemma 4 (Preciseness for acyclic Kripke structures). Let $K = (V, E, I, L)$ be a Kripke structure, where E constitutes a directed acyclic graph, let $i, j \in V$ be two situations and $m \in \text{Mset}$ a multiset with dom $m = V$. Then

$$\forall tr \in R_{i,j,m} : tr \models_K \phi \quad \text{iff} \quad \texttt{true} \in K\llbracket \phi \rrbracket_c \, i\, j\, m \ .$$

Uniqueness of the m-consistent path, however, is a sufficient yet not necessary condition for preciseness. As between chop points, paths visiting the same states in different sequence are indistinguishable, counting visits to all basic blocks of a Kripke structure suffices to determine satisfaction of a formula for all paths consistent with the count, provided that chop and conjunction occur in restricted polarity. Note that it is to be expected that we have preciseness for restricted classes only, because precise approximation for arbitrary formulas and Kripke structures would conflict with our goal of providing a decidable model checking problem and an elementary model checking procedure.

Hereby, a *basic block* is a connected component of the graph that remains after eliminating all incoming edges to vertices having more than one incoming edge or being initial and additionally having an incoming edge (i.e., all joins) and all outgoing edges of vertices having more than one outgoing edge (i.e., all forks). Note that except for unreachable cycles, the (transitive closure of the) edge relation in the graph pruned according to these rules constitutes a linear order on the states in a basic block. Given a reachable basic block B of K, we denote by $\min B$ (or $\max B$, resp.) the unique minimal (maximal) element with respect to that order. We say that $W \subseteq V$ *covers the basic blocks of* K iff $V_B \cap W \neq \emptyset$ for each basic block $V_B \subset V$ of K.

Lemma 5 (Preciseness when counting basic blocks). Let $K = (V, E, I, L)$ be a Kripke structure, $i, j \in V$ two situations, and $m \in \mathrm{Mset}$ a multiset with $\operatorname{dom} m$ covering the basic blocks of K. Let ϕ be a formula where all occurrences of chop are under the same polarity and all occurrences of conjunction under the dual polarity, i.e. chops are either all in negative context and conjunctions in positive or all chops are in positive and all conjunctions in negative context. Then

$$\forall tr \in R_{i,j,m} : tr \models_K \phi \quad \text{iff} \quad \texttt{true} \in K[\![\phi]\!]_c\, i\, j\, m$$

whenever the occurrences of chop are in negative and the occurrences of conjunction in positive context and

$$\forall tr \in R_{i,j,m} : tr \not\models_K \phi \quad \text{iff} \quad \texttt{false} \in K[\![\phi]\!]_c\, i\, j\, m$$

whenever the occurrences of chop are in positive and those of conjunction in negative context.

Proof. By induction on the structure of the formula ϕ. The induction has two base cases:

1. $\phi = \top$: The conjecture holds, as $tr \models_K \top$ for each tr and as $\texttt{true} \in K[\![\top]\!]_c\, i\, j\, m$, and because $\texttt{false} \in K[\![\top]\!]_c\, i\, j\, m$ implies $R_{i,j,m} = \emptyset$.

2. $\phi = \Sigma_{i\in\Omega} c_i \int S_i \bowtie k$: As the basic blocks are branch- and join-free, m together with i, j fully determines the frequency of visit to each individual vertex in V when m covers the basic blocks of K. Thus, $\Sigma_{i\in\Omega} c_i \int S_i$ is the same for all m-consistent traces from i to j in K. Consequently, either $R_{i,j,m} = \emptyset$, in which case the counting semantics assigns $\{\texttt{true}, \texttt{false}\}$ consistently with the trivially satisfied fact that $\forall tr \in R_{i,j,m} : tr \models_K \phi$ as well as $\forall tr \in R_{i,j,m} : tr \not\models_K \phi$, or $R_{i,j,m} \neq \emptyset$ and $\forall tr \in R_{i,j,m} : tr \models_K \phi$, in which case the counting semantics correctly assigns $\{\texttt{true}\}$ according to its definition, or $R_{i,j,m} \neq \emptyset$ and $\forall tr \in R_{i,j,m} : tr \not\models_K \phi$, in which case the counting semantics correctly assigns $\{\texttt{false}\}$.

For the induction step, the case $\phi = \neg\phi_1$ is straightforward taking into account that the polarities swap, and the case $\phi = \phi_1 \wedge \phi_2$ is easy because the conjunction has positive polarity here.

For $\phi = \phi_1 \frown \phi_2$, we observe that chop occurs in positive context here and we thus have to prove that

$$\forall tr \in R_{i,j,m} : tr \not\models_K \phi \quad \text{iff} \quad \mathtt{false} \in K[\![\phi]\!]_c\, i\, j\, m.$$

As the implication from right to left is covered by Lemma 3, it remains to be shown that $\forall tr \in R_{i,j,m} : tr \not\models_K \phi$ implies $\mathtt{false} \in K[\![\phi]\!]_c\, i\, j\, m$. As the latter is trivially satisfied by definition of the counting semantics whenever $R_{i,j,m} = \emptyset$, we assume in the remainder that $R_{i,j,m}$ is non-empty and $tr \not\models_K \phi$ for each $tr \in R_{i,j,m}$.

Let $k \in V_{i,j,m}$ and m_1, m_2 be a consistent decomposition of m wrt. i, j, k. For each consistent decomposition m_1, m_2 of m, the multisets m_1 and m_2 both cover the basic blocks of K if m does. Hence, we know from the induction hypothesis that

$$\forall tr \in R_{i,k,m_1} : tr \not\models_K \phi_1 \quad \text{iff} \quad \mathtt{false} \in K[\![\phi_1]\!]_c\, i\, k\, m_1$$

as well as

$$\forall tr \in R_{k,j,m_2} : tr \not\models_K \phi_2 \quad \text{iff} \quad \mathtt{false} \in K[\![\phi_2]\!]_c\, k\, j\, m_2 \ .$$

This implies that $\mathtt{false} \in K[\![\phi_1]\!]_c\, i\, k\, m_1$ or $\mathtt{false} \in K[\![\phi_2]\!]_c\, k\, j\, m_2$, as otherwise there were $tr_1 \in R_{i,k,m_1}$ and $tr_2 \in R_{k,j,m_2}$ such that $tr_1 \models_K \phi_1$ and $tr_2 \models_K \phi_2$, i.e. $tr_1 \frown tr_2 \models_K \phi_1 \frown \phi_2$ and $tr_1 \frown tr_2 \in R_{i,j,m}$, contradicting the assumption that $tr \not\models_K \phi$ for each $tr \in R_{i,j,m}$.

Thus, $\mathtt{false} \in K[\![\phi_1]\!]_c\, i\, k\, m_1$ or $\mathtt{false} \in K[\![\phi_1]\!]_c\, k\, j\, m_2$ for each $k \in V_{i,j,m}$ and each consistent decomposition m_1, m_2 of m wrt. i, j, k, which implies $\mathtt{false} \in K[\![\phi]\!]_c\, i\, j\, m$ by the definition of the counting semantics. □

We say that a multiset m_2 is *more defined* than a multiset m_1 (written $m_1 \sqsubseteq m_2$), if the domain of m_1 is included in the domain of m_2 and they agree on vertices for which they are both defined, i.e. $\operatorname{dom} m_1 \subseteq \operatorname{dom} m_2$ and for all $v \in \operatorname{dom} m_1$ we have that $m_1(v) = m_2(v)$.

For the *fully undefined multiset* $\bot_{\mathcal{M}}$ (with empty domain), we have that every run is $\bot_{\mathcal{M}}$-consistent, and in this case, the counting semantics will degenerate to the doubly situation-based semantics as stated in the following lemma.

Lemma 6. Let $K = (V, E, I, L)$ be a Kripke structure and $i, j \in V$. Then

$$K[\![\phi]\!]_c\, i\, j\, \bot_{\mathcal{M}} = K[\![\phi]\!]\, i\, j \ .$$

Furthermore, if m_2 is more defined than m_1, then an m_2-consistent run is also m_1-consistent, and, therefore, more defined multisets yield more precise approximations.

Lemma 7 (Refinement). Let $K = (V, E, I, L)$ be a Kripke structure, $i, j \in V$, and $m_1, m_2 \in \text{Mset}$. Then

$$m_1 \sqsubseteq m_2 \implies K[\![\phi]\!]_c\, i\, j\, m_1 \subseteq K[\![\phi]\!]_c\, i\, j\, m_2 \,.$$

Notice that $\emptyset$ gives least precision and $\{\mathtt{false}, \mathtt{true}\}$ represents a contradiction.

4.1 A integer-linear programming formulation of the consistency condition

Let $K = (V, E, I, L)$ be a Kripke structure and m a partial multiset covering the basic blocks of K. Any covering of the basic blocks will do, but we will choose one consisting just of the maximal elements of the basic blocks, i.e.

$$\text{dom } m = \{\max B \mid \text{where } B \text{ is a basic block of } K\}$$

We denote by v_B the representative of the basic block B in dom m.

For each vertex $v_B \in$ dom m we introduce a variable named $m[v_B]$. Let $\overline{m}$ denote the vector of variable $m[v_{B_1}], m[v_{B_2}] \dots m[v_{B_n}]$, where domain dom $m = \{v_{B_1}, v_{B_2}, \dots, v_{B_n}\}$. Furthermore, for each edge $(v, w) \in E$ in the Kripke structure which is not an edge inside any basic block, i.e. only for edges transferring control between basic blocks, we introduce a variable $e[v, w]$ counting the number of times control is transferred via edge (v, w).

We shall give a system of linear constraints having $\overline{m}$ and $\overline{e}$ as the only free variables, named $C(K, i_0, j_0, \overline{m}, \overline{e})$, so that m-consistency wrt. K, i_0 and j_0 is equivalent to satisfiability of $C(K, i_0, j_0, \overline{m}, \overline{e})$.

To achieve this, we extend the notation $m[v]$ to all vertices $v \in V$ without introducing new variables beyond $\overline{m}$. Instead, we alias appropriate variables in $\overline{m}$ as follows. For each basic block $V_B \subset V$ and each $v \in V_B \setminus \text{dom}\, m$, let

$$m[v] = m[v_B] + I_v + J_v \ ,$$

where

$$I_v = \begin{cases} -1 & \text{if } v \text{ comes strictly before } i_0 \\ 0 & \text{otherwise,} \end{cases}$$

$$J_v = \begin{cases} 1 & \text{if } v \text{ comes strictly before } j_0, \\ 0 & \text{otherwise.} \end{cases}$$

where "comes strictly before" is understood wrt. to the control flow in the basic block.

Note that above $m[v]$ can be used as a symbolic term determining the visiting frequency of unaccounted vertices v without introducing further variables $m[v]$, i.e. without extending the multiset.

The constraints (defined in the following) express informally that the "inflow" is the same as the "outflow" for any basic block V'. The start and end vertices i_0 and j_0, respectively, get special treatment to express that i_0 has an extra outflow of 1, and that the last visit to j_0 is not counted in the multiset $\overline{m}$. We assume below that i_0 and j_0 are in different basic blocks. The equations for the case where they are in the same block are obtained by a straightforward revision taking into account the mutual position of i_0 and j_0 in the block.

For every basic block B neither containing the start vertex i_0 nor the end vertex j_0, i.e. $i_0, j_0 \notin V_B$, there are two equations expressing that the frequency of visiting the start vertex $v_s = \min B$ of the block coincides with the number of entries to B as well as with the number of exits from the last vertex v_B of B:

$$\Sigma_{(l,v_s)\in E}\, e[l, v_s] = m[v_s] \quad \text{and} \quad m[v_B] = \Sigma_{(v_B,l)\in E}\, e[v_B, l] \tag{1}$$

where $v_s = \min B$ and $l \in V$ is arbitrary. Note that due to the aliasing rules $m[v_s]$ is identical to $m[v_B]$ when i_0 and j_0 are not elements in the block.

In order to characterize runs from i_0 to j_0, we give i_0 one visit extra beyond its inflow through edges and analogously j_0 one visit less than its inflow (the last visit to the terminal edge is not counted in m, as it does not contribute to accumulated durations). For all the other vertices, we keep in- and outflow balanced and let them agree with the number of visits.

Assume that i_0 is in basic block B (and that j_0 is not in that block). Let $v_s = \min B$ be the first vertex and v_B the last vertex of B as before. The equations for the block containing the start vertex are:

$$\begin{aligned} 1 + \Sigma_{(k,v_s)\in E} e[k, v_s] &= m[i_0] \\ m[v_B] &= \Sigma_{(v_B,k)\in E} e[v_B, k] \end{aligned} \tag{2}$$

Assume now that j_0 is in basic block B (and that i_0 is not in that block). Then the equations for the block containing the end vertex are:

$$\begin{aligned} \Sigma_{(k,v_s)\in E} e[k, v_s] &= m[j_0] + 1 \\ m[v_B] &= \Sigma_{(v_B,k)\in E} e[v_B, k] \end{aligned} \tag{3}$$

The constraints presented so far, which contain 2 equations per basic block, constitutes a *necessary* condition for consistency. I.e., for every trace from i_0 to j_0, there is a corresponding satisfying assignment to the above constraints. But the converse is not necessarily true as the simple example in Fig. 3 shows. It is easy to see that the fixed-point-equations arising permit assigning uniform visiting frequencies to a cycle without actually reaching it.

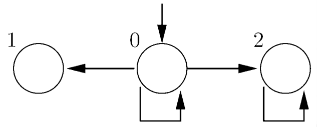


Figure 3. A Kripke structure

The constraint system for the case where the starts vertex is $i_0 = 0$, the end vertex is $j_0 = 1$, and the domain of the multiset is $\operatorname{dom} m = \{0, 1, 2\}$ due to all vertices forming atomic basic blocks is:

$$\begin{aligned} 1 + e[0,0] &= m[0] \\ m[0] &= e[0,0] + e[0,2] + e[0,1] \\ e[0,1] &= m[1] + 1 \\ m[1] &= 0 \\ e[0,2] + e[2,2] &= m[2] \\ m[2] &= e[2,2] \end{aligned}$$

These equations express that

1. vertex 0 is visited once more than entered through edges, as it is initial due to $i_0 = 1$,

2. the outflow through the three outgoing edges departing from 0 coincides with the frequency of visiting 0,
3. the visit frequency of vertex 1 is one less than the frequency of entering edges taken, as the last visit to 1 is not accounted for due to $j_0 = 1$,
4. the number of visits to 1 accounted for (i.e., excluding the final visit) coincides to the number of leaves via an outgoing edge, which is zero due to no outgoing edges being present,

to explain just the first four.

These first four equations imply that $e[0,2] = 0$, which is meaningful since there can be no transition from vertex 0 to vertex 2 for a trace starting in vertex 0 and ending in vertex 1. But this means that the only constraint on $e[2,2]$ and $m[2]$ remaining is

$$e[2,2] = m[2] \ ,$$

allowing satisfying assignments which do not correspond to any trace from vertex 0 to vertex 1, like the assignment $m[0] = 2, m[1] = 0$ and $m[2] = 1000$.

To avoid such undesirable assignments, we add the additional constraints that a loop has to be entered to have a non-zero visit frequency. Loops involving the start vertex i_0 can be disregarded as the start vertex by default is entered.

For simplicity, we assume that the loops are structured and, consequently, have exactly one entry point v_s which by definition is the start vertex of some basic block, thus being covered by vector $\overline{e}$. As the frequency of visiting v_s (and all subsequent vertices in the loop) is zero unless the loop is explicitly entered by either i_0 being in the loop or via an edge going into v_s from outside the loop, we add the following constraint for each start vertex v_s of a loop not containing i_o:

$$m[v_s] > 0 \implies \sum_{(v',v)\in E', v' \text{ outside and } v \text{ inside the loop}} e[v',v] > 0 \qquad (4)$$

For the simple example in Fig. 3 only the self-loop in vertex 2 is relevant, as the other loop involves the start vertex, and we generate one constraint in this case:

$$m[2] > 0 \implies e[0,2] > 0.$$

Adding the extra constraints (4) we have achieved a necessary and sufficient consistency condition. Let $C(K, i_0, j_0, \overline{m}, \overline{e})$ denote the conjunction of the above constraints (1)-(4). Given $n \in \text{Mset}$, we write $n \models C(K, i, j, \overline{m}, \overline{e})$ if $C(K, i_0, j_0, \overline{m}, \overline{e})$ after, for each $k \in \operatorname{dom} m$, substituting $n(k)$ for $m[k]$ is satisfiable.

Lemma 8. Let $n \in \text{Mset}$ with $\operatorname{dom} n \supseteq \operatorname{dom} m$.

$$n \text{ is consistent wrt. } K, i, j \text{ if and only if } n \models C(K, i, j, \overline{m}, \overline{e}).$$

Proof. The implication: n is consistent wrt. K, i, j implies $n \models C(K, i, j, \overline{m}, \overline{e})$, is not difficult to establish. Given an n-consistent trace from i to j is it easy to verify that the constraints in $C(K, i, j, \overline{m}, \overline{e})$ are satisfied.

For the other direction assume that $n \models C(K, i_0, j_0, \overline{m}, \overline{e})$. We must show that there is an n-consistent trace from i_0 to j_0. The main idea is to remove "active loops" in K in a "well-chosen way" resulting in a new assignment n' satisfying that there is no active loop in K given n'. Furthermore, the loop removals preserve the constraints $C(K, i_0, j_0, \overline{m}, \overline{e})$ and the property that there is an n-consistent trace from i_0 to j_0 iff there is an n'-consistent trace from i_0 to j_0.

This idea is now formalized for a given Kripke structure K. We start with the following definitions:

- By abuse of notation, we let $\overline{m}$ and $\overline{e}$ together denote a satisfying assignment for $C(K, i_0, j_0, \overline{m}, \overline{e})$.
- A sequence of vertices $v_{s_1} v_{s_2} \ldots v_{s_k} v_{s_1}$, where each v_{s_i} is the start vertex of some basic block with representative v_{B_i}, is called an *active loop* in K given assignment $\overline{m}$ and $\overline{e}$, if $\overline{e}[v_{B_i}, v_{s_{i+1}}] \geqslant 1$, for $1 \leqslant i < k$, and $\overline{e}[v_{B_k}, v_{s_1}] \geqslant 1$. By the *loop frequency* we shall understand the minimal frequency on the edges of the loop, i.e. $f = \min(\{\overline{e}[v_{B_i}, v_{s_{i+1}}] \mid 1 \leqslant i < n\} \cup \{\overline{e}[v_{B_k}, v_{s_1}]\})$.
- Let $v_{s_1} v_{s_2} \ldots v_{s_k} v_{s_1}$, be an *active loop* in K with frequency f given assignment $\overline{m}$ and $\overline{e}$. By *removal* of that active loop, we shall understand a new assignment $\overline{m'}$ and $\overline{e'}$ defined as follows:

$$\overline{m'}(v) = \begin{cases} \overline{m}(v) - f, & \text{if } v \text{ is a part of the loop} \\ \overline{m}(v), & \text{otherwise} \end{cases}$$

$$\overline{e'}[v, v'] = \begin{cases} \overline{e}[v, v'] - f, & \text{if the edge } (v, v') \text{ is a part of the loop} \\ \overline{e}[v, v'], & \text{otherwise} \end{cases}$$

- A *guard* of an active loop $v_{s_1} v_{s_2} \ldots v_{s_k} v_{s_1}$ in K given assignment $\overline{m}$ and $\overline{e}$ is an edge $(v, v_{s_i}) \in E$ for which v_{s_i} is some loop vertex, v does not belong to any basic block in the loop and $\overline{e}[v, v_{s_i}] \geqslant 1$.

Notice that the consistency constraints (4) guarantee that each active loop has at least one guard.

It is not difficult to check that loop removal preserves the flow-constraint part of the consistency condition, i.e. equations (1)-(3), in the sense that if $\overline{m}$ and $\overline{e}$ satisfy these constraints, then so will $\overline{m'}$ and $\overline{e'}$ do. But in general, the last type of constraints (4) will not be preserved. The problem is that removal of one loop may result in that another loop will become unreachable because all guards to that loop are removed. We shall devise a safe method of removing loops while preserving all consistency constraints.

Let L_1 and L_2 two active loops given $\overline{m}$ and $\overline{e}$. L_1 *subsumes* L_2 if

- the guards of L_2 belong to the loop L_1 and
- L_1 is the only loop which edges of L_2 are guarding.

If L_1 subsumes L_2, then L_2 can be safely removed. To see this, notice first that the structured loop consisting of L_1 and L_2 is active as well, and must have an external guard, which must be a guard of L_1 alone as every guard of L_2 belongs to L_1. Therefore, if $\overline{m'}$ and $\overline{e'}$ result from removing L_2, the constraint (4) is preserved (and the others as well) as the only involved loop L_1 still has an active guard.

Let v be a vertex and L an active loop including v given $\overline{m}$ and $\overline{e}$. v *subsumes* L if

- All edges in L and vertices in L except v have the same frequency f and
- v has a visiting frequency which exceeds f.

If v subsumes L, then L can safely be removed since the edges of L do not act as guards.

Suppose that no subsumed loop exists given $\overline{m}$ and $\overline{e}$. Consider an arbitrary active basic loop L, i.e. a loop with no repetitions except for the start and end vertex. Every active loop has guards from two different loops as otherwise it would be a subsumed loop. Therefore, L can be safely removed.

It is not difficult to see that each of these three rules for safe loop removal preserves the trace property: there is a consistent trace from i_0 to j_0 given $\overline{m'}$ and $\overline{e'}$ iff there is a consistent trace from i_0 to j_0 given $\overline{m}$ and $\overline{e}$.

The safe loop removal method is now to repeat the following two steps until all active loops are removed:

1. Removed all subsumed active loops.
2. Remove an arbitrary active basic loop, when no subsumed active loop exists.

The resulting assignment $\overline{m'}$ and $\overline{e'}$ satisfies the consistency constraints and will define a unique path from i_0 to j_0 since the flow equations hold, i_0 has an extra inflow of 1, j_0 has an extra outflow of 1, and each vertex has an outflow of at most 1, as otherwise there would be an active loop. There can be no isolated sequential components, since that would imply the existence of a vertex with a net inflow of 0 and a positive visiting frequency and this would violate the constraints. $\square$

Satisfiability of $C(K,i,j,\overline{m},\overline{e})$ is a necessary and sufficient condition for existence of an m-consistent trace from i to j. Note that the formula $C(K,i,j,\overline{m},\overline{e})$ has size linear in $|V|+|E|$ and can be generated in $O(|V|+|E|)$ time.

4.2 Model checking

We now present an algorithm for checking whether a given Kripke structure $K=(V,E,I,L)$ satisfies a formula ϕ wrt. the counting semantics. It consists of, first, an exponential (in the chop depth) labelling phase which generates a side condition in the form of a Presburger formula, i.e. a quantified Boolean combination of integer linear constraints and, second, the satisfiability check of this condition, which can be done in triple exponential time. For formulas ϕ having chops in negative polarity only, the formulas to be checked for satisfiability are quantifier free and can be checked in NP using a decision algorithm for integer LinSAT, i.e. quantifier-free Boolean combinations of linear constraints over the integers, such as SAT-modulo-theory over linear integer arithmetic[19].

Phase 1: Labelling state pairs

The input to this phase consists of the Kripke structure K, the formula ϕ and the domain of the multiset dom $m=\{v_1,\ldots,v_n\}$, where we use the same domain as in the construction of the consistency formula $C(K,i,j,\overline{m},\overline{e})$.

This phase of the algorithm proceeds by induction on the syntactic structure of ϕ, marking situation pairs $(i,j)\in V\times V$ with triplets $(\psi,b,lin(\overline{m},\overline{e}))$ of a subformula ψ of ϕ, a Boolean value b, and an integer linear satisfiability problem $lin(\overline{m},\overline{e})$, having the vectors $\overline{m}$ and $\overline{e}$ as the only free variables. A marking $(\psi, b, lin(\overline{m}, \overline{e}))$ of (i, j) denotes that:

$$\{b\} = K[\![\psi]\!]_c \, i \, j \, m$$

for all multi-sets m with dom $m = \{v_1, \ldots, v_n\}$ satisfying $m \models lin(\overline{m}, \overline{e})$.

For each state pair (i, j) and each subformula ψ of ϕ, we provide two markings $(\psi, \texttt{true}, lin_t(\overline{m}, \overline{e}))$ and $(\psi, \texttt{false}, lin_f(\overline{m}, \overline{e}))$.

1. *Base case* $\phi = \top$. As the formula $\top$ is satisfied by all consistent triplets (i, j, m), all state pairs become marked with $(\top, \texttt{true}, C(K, i, j, \overline{m}, \overline{e}))$ and $(\top, \texttt{false}, \texttt{false})$. Given the $O(|V| + |E|)$ size of the formula $C(K, i, j, \overline{m}, \overline{e})$, these markings can be performed in $O(|V|^2(|V| + |E|))$ time.

2. *Base case* $\phi = \Sigma_{i\in\Omega} c_i \int S_i < k$. The other cases of $\Sigma_{i\in\Omega} c_i \int S_i \bowtie k$ are similar. As m covers the basic blocks and m has been (symbolically) extended to all vertices, we can assign conditions for satisfaction or violation by a multi-set m to all pairs (i, j) within one sweep over $V \times V$ as follows: mark (i, j) with

   - $\left(\Sigma_{i\in\Omega} c_i \int S_i < k, \texttt{true}, C(K, i, j, \overline{m}, \overline{e}) \wedge \sum_{i\in\Omega} c_i \sum_{v\in V, v\models S_i} m[v] < k\right)$ and
   - $\left(\Sigma_{i\in\Omega} c_i \int S_i < k, \texttt{false}, C(K, i, j, \overline{m}, \overline{e}) \wedge \sum_{i\in\Omega} c_i \sum_{v\in V, v\models S_i} m[v] \geqslant k\right)$.

   The complexity of this is $O(|V|^2(|V| + |E|))$ due to the size of $C(K, i, j, \overline{m}, \overline{e})$. This means that the overall time complexity of marking formula $\Sigma_{i\in\Omega} c_i \int S_i < k$ is $O(|V|^2(|V| + |E|))$. The generated individual formulas are again of size $O(|V| + |E|)$.

3. *Case* $\phi = \neg\psi$. We assign markings for $\neg\psi$ to all pairs (i, j) within one sweep over $V \times V$ as follows:

   - mark (i, j) with $(\neg\psi, \texttt{true}, \mu)$ iff (i, j) is marked with $(\psi, \texttt{false}, \mu)$, and
   - mark (i, j) with $(\neg\psi, \texttt{false}, \nu)$ iff (i, j) is marked with $(\psi, \texttt{true}, \nu)$.

   With appropriate data structures supporting the sharing of common (sub-)formulas, the time complexity of this step is $O(|V|^2)$.

4. *Case* $\phi = \chi \wedge \psi$. We assign markings for $\chi \wedge \psi$ to all pairs (i, j) within one sweep over $V \times V$ as follows:

   - mark (i, j) with $(\chi \wedge \psi, \texttt{true}, \mu \wedge \nu)$ iff (i, j) is marked with $(\chi, \texttt{true}, \mu)$ and with $(\psi, \texttt{true}, \nu)$, and
   - mark (i, j) with $(\chi \wedge \psi, \texttt{false}, \mu \vee \nu)$ iff (i, j) is marked with $(\chi, \texttt{false}, \mu)$ and with $(\psi, \texttt{false}, \nu)$.

   Using appropriate data structures, the time complexity of this step is again $O(|V|^2)$.

5. *Case* $\phi = \chi \frown \psi$. We assign markings for $\chi \frown \psi$ to all pairs (i, j) as follows:

- mark (i,j) with $(\chi \frown \psi, \texttt{true}, lin_t(\overline{m},\overline{e}))$ for

  $$lin_t(\overline{m},\overline{e}) \;=\; \bigvee_{k\in V} \left(\begin{array}{l} \exists \overline{m}_1, \overline{m}_2, \overline{e}_1, \overline{e}_2 : \mu \\ \wedge \\ \forall \overline{m}_1, \overline{m}_2, \overline{e}_1, \overline{e}_2 : \mu \Rightarrow (\mu_k[\overline{m}_1, \overline{e}_1/\overline{m},\overline{e}] \wedge \nu_k[\overline{m}_2, \overline{e}_2/\overline{m},\overline{e}]) \end{array} \right),$$

  $$\mu \;=\; \bigwedge_{v,w\in \mathrm{dom}\ m} \left(\begin{array}{ll} & m[v] = m_1[v] + m_2[v] \\ \wedge & e[v,w] = e_1[v,w] + e_2[v,w] \\ \wedge & C(K,i,k,\overline{m_1},\overline{e}_1) \wedge C(K,k,j,\overline{m_2},\overline{e}_2) \end{array} \right),$$

  where $\overline{m}_l = m_l[v_1],\ldots,m_l[v_n]$ and $\overline{e}_l = e_l[v_1,w_1],\ldots,e_l[v_n,w_n]$, for $l \in \{1,2\}$, are four vectors of fresh variables, (i,k) is marked with $(\chi, \texttt{true}, \mu_k)$ and (k,j) with $(\psi, \texttt{true}, \nu_k)$. We use $\xi\,[y_1,\ldots,y_n/x_1,\ldots,x_n]$ to denote the formula obtained from ξ by substituting every free occurrence of x_j with y_j, for $1 \leqslant j \leqslant n$.
- mark (i,j) with $(\chi \frown \psi, \texttt{false}, lin_f(\overline{m},\overline{e}))$ for

  $$lin_f(\overline{m},\overline{e}) \;=\; C(K,i,j,\overline{m},\overline{e}) \wedge \bigwedge_{k\in V} \left(\forall \overline{m}_1, \overline{m}_2, \overline{e}_1, \overline{e}_2 : \mu \Rightarrow (\mu_k[\overline{m}_1, \overline{e}_1/\overline{m},\overline{e}] \vee \nu_k[\overline{m}_2, \overline{e}_2/\overline{m},\overline{e}]) \right)$$

  where μ is as in the previous case, (i,k) is marked with $(\chi, \texttt{false}, \mu_k)$ and (k,j) with $(\psi, \texttt{false}, \nu_k)$.

Note that this step, in contrast to the previous steps which generate markings linear in the size of the DC formula, yields a $|V|$-fold blow-up in formula size, which thus dominates the space complexity of the whole procedure. The size of the formulas generated is $O(u(|V|+|E|)|V|^c)$, where c is the chop nesting depth of ϕ and u the nesting depth of conjunctions.

This algorithm recurs over all subformulas of ϕ in order to determine the situation pairs satisfying ϕ.

Lemma 9 (Correctness of marking algorithm). Let $K = (V,E,I,L)$ be a Kripke structure, $i,j \in V$, ϕ a DC formula, and m a multiset. Let $(\phi, \texttt{true}, lin_t(\overline{m},\overline{e}))$ and $(\phi, \texttt{false}, lin_f(\overline{m},\overline{e}))$ be the (unique) markings obtained for formula ϕ on (i,j) using the domain of m. Then

$$\{\texttt{true}\} = K[\![\phi]\!]_c\, i\, j\, m \quad \text{iff} \quad m \models lin_t(\overline{m},\overline{e}) \qquad (a)$$
$$\{\texttt{false}\} = K[\![\phi]\!]_c\, i\, j\, m \quad \text{iff} \quad m \models lin_f(\overline{m},\overline{e}) \qquad (b)$$

Proof. The proof is by induction on the structure of ϕ.

The base cases and the two recursive cases $\phi = \neg\psi$ and $\phi = \chi \wedge \psi$ are simple.

For the correctness of the markings computed for chop, observe that these markings try all possible chop points as well as all possible decompositions of the original multiset into two parts, thus checking all consistent decompositions as required by the counting semantics. The correctness of the marking thus follows from the induction hypothesis. □

Corollary 1. Let $K = (V, E, I, L)$ be a Kripke structure, $i, j \in V$, and ϕ a DC formula. Let $(\phi, \texttt{true}, lin_t)$ be a marking for the formula ϕ on (i, j) for some domain $V' \subseteq V$ of a multiset. Then validity of $C(K, i, j, \overline{m}, \overline{e}) \Rightarrow lin_t$ over the integers implies that $tr \models_K \phi$ holds for each run tr from i to j.

Proof. Assume that $tr \not\models \phi$ for some run tr of K from i to j. The trace tr induces at assignments for the two vector of variables $\overline{m}$ and $\overline{e}$, for which $C(K, i, j, \overline{m}, \overline{e})$ is true according to Lemma 8. Furthermore, $K[\![\phi]\!]_c\, i\, j\, m \neq \{\texttt{true}\}$ according to Lemma 3 and, therefore, according to Lemma 9: $m \not\models lin_t$. Hence, $C(K, i, j, \overline{m}, \overline{e}) \Rightarrow lin_t$ is invalid. □

Phase 2: Deciding the Presburger side condition

When checking a formula ϕ, the marking phase yields a marking $(\phi, \texttt{true}, \mu_{i,j})$ for each state pair (i, j). The next step is to check the linear formulas $C(K, i, j, \overline{m}, \overline{e}) \Rightarrow \mu_{i,j}$ for validity:

6. For each initial vertex $i \in I$ and each $j \in V$ decide the satisfiability of the formula $C(K, i, j, \overline{m}, \overline{e}) \wedge \neg\mu_{i,j}$, where $(\phi, \texttt{true}, \mu_{i,j})$ is the marking of (i, j) from steps 1 to 5. Satisfiability can be decided using any decision procedure for Presburger arithmetic. If all these formulas are unsatisfiable then report "$K \models \phi$".

The size of $\mu_{i,j}$ is $O(u(|V| + |E|)|V|^c)$, where c is the chop depth and u the depth of conjunctions in ϕ. As the $\mu_{i,j}$ are formulas of Presburger arithmetic, these can be checked in triple exponential time rendering the overall algorithm 4-fold exponential.

The following theorem is a consequence of Corollary 1.

Theorem 2. Above model checking algorithm is sound. I.e., if it reports "$K \models \phi$" in step 6 then K satisfies ϕ.

The algorithm based on the above marking can be simplified, knowing that we should check formulas of the form $C(K, i, j, \overline{m}, \overline{e}) \wedge \neg\mu_{i,j}$ for satisfiability. In particular, we can make make Boolean simplification as well as quantifier eliminations. Let $lin_t(\phi, i, j, \overline{m}, \overline{e})$ and $lin_f(\phi, i, j, \overline{m}, \overline{e})$ be the true and false markings of ϕ for the situation pair (i, j). We shall devise a marking algorithm containing simpler formulas $sim_t(\phi, i, j, \overline{m}, \overline{e})$ and $sim_f(\phi, i, j, \overline{m}, \overline{e})$ so that

$$m \models C(K, i, j, \overline{m}, \overline{e}) \wedge \neg lin_b(\phi, i, j, \overline{m}, \overline{e}) \text{ iff } m \models C(K, i, j, \overline{m}, \overline{e}) \wedge \neg sim_b(\phi, i, j, \overline{m}, \overline{e})$$

for $b \in \{t, f\}$.

The simplified formulas are defined as follows:

$$\begin{aligned}
sim_t(\top, i, j, \overline{m}, \overline{e}) &= \texttt{true} \\
sim_f(\top, i, j, \overline{m}, \overline{e}) &= \texttt{false} \\
sim_t(\Sigma_{i\in\Omega} c_i \textstyle\int S_i < k, i, j, \overline{m}, \overline{e}) &= \textstyle\sum_{i\in\Omega} c_i \sum_{v\in V, v\models S_i} m[v] < k \\
sim_f(\Sigma_{i\in\Omega} c_i \textstyle\int S_i < k, i, j, \overline{m}, \overline{e}) &= \textstyle\sum_{i\in\Omega} c_i \sum_{v\in V, v\models S_i} m[v] \geqslant k
\end{aligned}$$

$$\begin{aligned} sim_t(\neg\phi, i, j, \overline{m}, \overline{e}) &= sim_f(\phi, i, j, \overline{m}, \overline{e}) \\ sim_f(\neg\phi, i, j, \overline{m}, \overline{e}) &= sim_t(\phi, i, j, \overline{m}, \overline{e}) \\ sim_t(\phi_1 \wedge \phi_2, i, j, \overline{m}, \overline{e}) &= sim_t(\phi_1, i, j, \overline{m}, \overline{e}) \wedge sim_t(\phi_2, i, j, \overline{m}, \overline{e}) \\ sim_f(\phi_1 \wedge \phi_2, i, j, \overline{m}, \overline{e}) &= sim_f(\phi_1, i, j, \overline{m}, \overline{e}) \vee sim_f(\phi_2, i, j, \overline{m}, \overline{e}) \end{aligned}$$

and

$$\begin{aligned} & sim_t(\phi_1 \frown \phi_2, i, j, \overline{m}, \overline{e}) \;=\\ &\qquad \bigvee\nolimits_{k \in V} \left(\begin{array}{l} \exists \overline{m}_1, \overline{m}_2, \overline{e}_1, \overline{e}_2 : \mu \\ \wedge \\ \forall \overline{m}_1, \overline{m}_2, \overline{e}_1, \overline{e}_2 : \mu \Rightarrow (sim_t(\phi_1, i, k, \overline{m}_1, \overline{e}_1) \wedge sim_t(\phi_2, k, j, \overline{m}_1, \overline{e}_2)) \end{array} \right) \\ & sim_f(\phi_1 \frown \phi_2, i, j, \overline{m}, \overline{e}) \;=\\ &\qquad \bigwedge\nolimits_{k \in V} \left(\; \mu \Rightarrow (sim_f(\phi_1, i, k, \overline{m}_1, \overline{e}_1) \vee sim_f(\phi_2, k, j, \overline{m}_2, \overline{e}_2)) \; \right) \end{aligned}$$

where

$$\mu \;=\; \bigwedge_{v,w \in \mathrm{dom}\ m} \left(\begin{array}{rl} & m[v] = m_1[v] + m_2[v] \\ \wedge & e[v,w] = e_1[v,w] + e_2[v,w] \\ \wedge & C(K, i, k, \overline{m_1}, \overline{e}_1) \wedge C(K, k, j, \overline{m_2}, \overline{e}_2) \end{array} \right)$$

Lemma 10 (Correctness of simplified markings).

$$\begin{aligned} & m \models C(K, i, j, \overline{m}, \overline{e}) \wedge \neg lin_b(\phi, i, j, \overline{m}, \overline{e}) \\ \text{iff} \quad & m \models C(K, i, j, \overline{m}, \overline{e}) \wedge \neg sim_b(\phi, i, j, \overline{m}, \overline{e}) \end{aligned}$$

for $b \in \{t, f\}$.

The proof is a simple inductive proof following the structure of formulas.

Step 6. for checking ϕ can now be improved by checking satisfiability of the formulas $C(K, i, j, \overline{m}, \overline{e}) \wedge \neg sim_t(\phi, i, j, \overline{m}, \overline{e})$ for every $i \in I$ and $j \in V$. If the chop-formulas of ϕ all occur in a negative polarity, it is easy to inspect that the formulas to be checked will be quantifier free and can be checked in NP using a decision algorithm for integer LinSAT, e.g. state of the art satisfiability-modulo-theory solvers supporting the theory of linear constraints over the integers, like Sateen[24], Z3[25], or YICES[26]. Since the generated formulas are of size exponential in the chop-depth, the algorithm is doubly exponential in the case where chop just occurs in negative polarity in ϕ.

Example: This model-checking algorithm can be used to establish $K \models \phi$, for $\phi = \Box(\ell < 4 \Rightarrow \int p < 3)$ and Kripke structure K from Fig. 2. Expanding the abbreviations in ϕ, we obtain the equivalent formula $\phi' = \neg((\top \frown (\int 1 < 4 \wedge \neg \int p < 3)) \frown \top)$. Table 2 shows the relevant markings generated by the model-checking algorithm. The Kripke structure K has three basic blocks. Vertices 1 and 2 belong to one block, where 1 is the start vertex and 2 the representative for that block, and vertices 3 and 4 constitute two singleton basic blocks. In the markings, we have omitted the vector $\overline{e}$ of edge variables and we have simplified the markings exploiting logical equivalences; but in some columns we have not expanded the abbreviation $m[1]$ for illustrative purposes. These simplifications render the tautology checks in step 6 of the algorithm trivial. The model-checking algorithm consequently reports "$K \models \Box(\ell < 4 \Rightarrow \int p < 3)$".

Table 2 Linear constraints assigned to state pairs (i, j) of the Kripke structure from Fig. 2 when checking $\phi = \Box(\ell < 4 \Rightarrow \int p < 3)$

i, j	$C(K, i, j, \overline{m}, \overline{e})$ after simplification	linear constraint η assigned to state pair i, j in marking (ψ, tv, η)					
		for $tv =$ `false` and $\psi =$				for $tv =$ `true` and $\psi =$	
		$\int\texttt{true} < 4$	$\neg\int p < 3$	ζ	$\Diamond\zeta$	$\int p < 3$	$\neg\Diamond\zeta$
1,1	$m[3] = m[4] = 0$	$m[1] > 2$	$m[1] < 3$	`true`	`true`	$m[1] < 3$	`true`
1,2	$m[3] = m[4] = 0$	$m[1] > 2$	$m[1] < 3$	`true`	`true`	$m[1] < 3$	`true`
1,3	$m[3] = m[4] = 0$	$m[1] > 1$	$m[1] < 3$	`true`	`true`	$m[1] < 3$	`true`
1,4	$m[2] > 0 \wedge m[3] = 1$	$\begin{pmatrix} m[1] > 1 \\ \vee m[4] > 0 \end{pmatrix}$	$m[1] \leqslant 1$	`true`	`true`	$m[1] \leqslant 1$	`true`
2,{1,3}	$\begin{pmatrix} m[3] = m[4] = 0 \\ \wedge m[2] > 0 \end{pmatrix}$	$m[1] > 1$	$m[1] < 2$	`true`	`true`	$m[1] < 2$	`true`
2,2	$m[3] = m[4] = 0$	$m[1] > 1$	$m[1] < 3$	`true`	`true`	$m[1] < 3$	`true`
2,4	$\begin{pmatrix} m[2] > 0 \\ \wedge m[3] = 1 \end{pmatrix}$	$\begin{pmatrix} m[1] > 0 \\ \vee m[4] > 1 \end{pmatrix}$	$m[1] = 0$	`true`	`true`	$m[1] = 0$	`true`
3,{1,2}	`false`	`true`	`true`	`true`	`true`	`true`	`true`
3,3	$\begin{pmatrix} m[2] = m[3] \\ = m[4] = 0 \end{pmatrix}$	`false`	`true`	`true`	`true`	`true`	`true`
3,4	$\begin{pmatrix} m[2] = 0 \\ \wedge m[3] = 1 \end{pmatrix}$	$m[4] \geqslant 3$	`true`	`true`	`true`	`true`	`true`
4,{1,2,3}	`false`	`true`	`true`	`true`	`true`	`true`	`true`
4,4	$m[2] = m[3] = 0$	$m[4] \geqslant 4$	`true`	`true`	`true`	`true`	`true`

The formulas occurring in the table are simplified. Second column: condition for path existence. Columns 3 to 6: markings $(\psi, \texttt{false}, \eta)$ generated by the algorithm for the different sub-formulas ψ occurring negatively in ϕ. The formula ζ abbreviates $\int 1 < 4 \wedge \neg\int p < 3$, i.e. $\neg\Diamond\zeta$ is the original formula ϕ. Columns 7: markings $(\psi, \texttt{true}, \eta)$ generated for positive sub-formulas, simplified. Column 8: the formula $C(K, i, j, \overline{m}, \overline{e}) \Rightarrow lin_t$ to be checked for validity, simplified. Hence, the algorithm answers "$K \models \phi$".

Notice that chop occurs in negative context in the formula ϕ and the doubly exponential algorithm applies in this case.

5 Conclusion

Aiming at efficient model checking by approximating its standard linear-time semantics, we have investigated two branching-time semantics for discrete-time DC, where intervals are generated from runs between two situations in a Kripke structure. Model checking is linear in the size of the formula and $O(|V|^3)$ in the size of the model for the first semantics, which unfortunately approximates linear time too coarsely to be useful. The second semantics extends the first with an occurrence count for crucial states, yielding a model-checking problem with a 4-fold exponential decision algorithm which approximates sufficiently accurately. For a restricted class of DC formulas the decision algorithm is doubly exponential.

Comparing this to the classical procedure proposed by Zhou, Hansen and Sestoft[10] and practically implemented by Skakkebæk and Sestoft[12] and by Pandya[14], it is not immediately obvious which procedure is more well-behaved in practice. The classical procedure maps a DC formula to an observer automaton of non-elementary size and then builds an automaton product with the Kripke structure, thus being non-elementary in the formula size yet linear in the size of the Kripke structure, while ours is doubly or even four-fold exponential in the Kripke structure also. In order to shed some light on this, we note the following facts:

1. In practice, building the observer automaton in the classical procedure often already fails for moderately sized formulas such that the procedure cannot even proceed to build the automaton product. The reason is that not only the alternation of chop and negation (which tends to be of rather low depth in practice) leads to explosion of the observer automaton, but also non-trivial time constants do, requiring a lot of counting in the observer. The counting structures in the automaton are exponential in the number of duration terms occurring in the formula, with the time constants as base. Note that our algorithm is less sensitive to time constants, as these are copied to the Presburger formulae representing the side conditions, where — at least in the quantifier-free case — they may be discharged by numerical procedures like integer linear programming.
2. The complexity figures stated for our procedure are extremely pessimistic in practice, as the procedure essentially collapses basic blocks to a single representative such that the number of basic blocks rather than the number of vertices determines the size of the formulas generated, which makes a tremendous difference in practice.
3. The classical procedure, being based on building an observer automaton, fails for the fragment of DC considered herein, as DC with weighted sums of durations is undecidable even in the discrete-time case.

Hence, it seems that the algorithms have a distinctly different scope. An interesting topic for future research could be whether there are useful combinations of the two.

Acknowledgements

We greatly appreciate comments from Kim G. Larsen and William Heise on earlier versions of this article.

References

[1] Zhou CC, Hoare CAR, Ravn AP. A calculus of durations. Information Processing Letters, 1991, 40(5): 269–276.

[2] Hansen MR, Zhou CC. Duration calculus: Logical foundations. Formal Aspects of Computing, 1997, 9(3): 283–330.

[3] Zhou CC, Hansen MR. Duration Calculus—A Formal Approach to Real-Time Systems. EACTS: Monographs in Theoretical Computer Science. Springer-Verlag, 2004.

[4] Fr anzle M. Model-checking dense-time duration calculus. Formal Aspects of Computing, 2004, 16(2): 121–139.

[5] Sharma B, Pandya PK, Chakraborty S. Bounded Validity Checking of Interval Duration Logic. In: Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2005). LNCS 3440, Springer-Verlag, 2005. 302–316.

[6] Zhou CC, Zhang JZ, Yang L, Li XS. Linear duration invariants. In: International Symposium on Formal Techniques in Real-Time and Fault-Tolerant systems (FTRTFT 1994). LNCS 863, Springer-Verlag, 1994. 86–109.

[7] Bouajjani A, Lakhnech Y, Robbana R. From duration calculus to linear hybrid automata. In: Computer Aided Verification (CAV 1995). LNCS 939, Springer-Verlag, 1995. 196–210.

[8] Laknech Y. Specification and Verification of Hybrid and Real-Time Sys-tems. Dissertation, Technische Fakultät der Christian-Albrechts-Universit at Kiel, 1996.

[9] Fränzle M, Hansen MR. Deciding an interval logic with accumulated durations. In: Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007). LNCS 4424, Springer-Verlag, 2007. 201–215.

[10] Zhou CC, Hansen MR, Sestoft P. Decidability and undecidability results for duration calculus. In: STACS 93. LNCS 665, Springer-Verlag, 1993. 58–68.

[11] Hansen MR. Model-checking discrete duration calculus. Formal Aspects of Computing, 1994, 6(6A): 826–845.

[12] Skakkebæk JU, Sestoft P. Checking validity of duration calculus formulas. ProCoS report ID/DTH JUS 3/1, Technical University of Denmark, 1994.

[13] Skakkebæk JU, Shankar N. Towards a duration calculus proof assistant in PVS. In: International Symposium on Formal Techniques in Real-Time and Fault-Tolerant systems (FTRTFT '94). LNCS 863, Springer-Verlag, 1994. 660–679.

[14] Pandya PK. Specifying and deciding quantified discrete-time duration calculus formulae using DCVALID. In: Workshop on Real-Time Tools, RT-TOOLS '2001. Aalborg, August 2001.

[15] Bolander T, Hansen JU, Hansen MR. Decidability of a hybrid duration calculus. Electronic Notes in Theoretical Computer Science, 2007, 174(6): 113–133.

[16] Pandya PK. Model Checking CTL*[DC]. In: Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001). LNCS 2031, Springer-Verlag, 2001. 559–573.

[17] Josko B. Modular Specification and Verification of Reactive Systems. Professorial dissertation, Universität Oldenburg, 1993.

[18] Fränzle M, Hansen MR. Efficient model checking for duration calculus based on branching-time approximations. In: Sixth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2008): IEEE, 2008. 63–72.

[19] Audemard G, Bertoli P, Cimatti A, Kornilowicz A, Sebastiani R. A SAT-based approach for solving formulas over boolean and linear mathematical propositions. In: Conference on Automated Deduction (CADE 18). LNAI 2392, Springer-Verlag, 2002. 193–208.

[20] Hansen MR, Hung DV. A Theory of Duration Calculus with Application. In: Domain Modelling and the Duration Calculus. LNCS 4710, Springer-Verlag, 2007. 119–176.

[21] Hansen MR. Duration Calculus. Chapter 6 of Logics of Specification Languages, EATCS: Monographs in Theoretical Computer Science. Springer-Verlag, 2008. 299–347.

[22] Moszkowski B. A temporal logic for multi-level reasoning about hardware. IEEE Computer, 1985, 18(2): 10–19.

[23] Zwick U. All pairs shortest paths using bridging sets and rectangular matrix multiplication. Journal of the ACM, 2002, 49.

[24] Kim H, Somenzi F, Jin H. Efficient Term-ITE conversion for satisfiability-modulo-theories. In: Theory and Applications of Satisfiability Testing (SAT 2009). LNCS 5584, Springer-Verlag, 2009. 195–208.

[25] de Moura L, Bjørner N. Z3: An Efficient SMT Solver. In: Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008). LNCS 4963, Springer-Verlag, 2008. 337–340.

[26] Dutertre B, de Moura L. A fast linear solver for DPLL(T). In: Computer Aided Verification (CAV 2006). LNCS 4144, Springer-Verlag, 2006. 81–94.