GKAT with Hoare Hypotheses

Jurriaan Rot
Radboud University, Nijmegen, The Netherlands

Todd Schmid
Bucknell University, Lewisburg, PA, USA

Jana Wagemaker
Radboud University, Nijmegen, The Netherlands

Abstract

Guarded Kleene Algebra with Tests (GKAT) is a variant of Kleene algebra which allows for reasoning about simple imperative programs, and which features a decision procedure for program equivalence in nearly linear time. In the current paper, we address the challenge of reasoning under assumptions about these programs. In particular, we develop a form of *Hoare hypotheses*, which allow modelling basic domain knowledge on pre- and postconditions of uninterpreted basic programs, and which are well-developed for classical Kleene algebra but not yet for GKAT. We show that the resulting axiomatisation is sound and complete. We then extend Hoare hypotheses to the more general form of *word hypotheses*. Based on an automata-theoretic approach, we show that equivalence of GKAT under word hypotheses is efficiently decidable.

2012 ACM Subject Classification Theory of computation → Semantics and reasoning; Theory of computation → Formal languages and automata theory

Keywords and phrases Kleene Algebra, GKAT, Hypotheses

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.46

Funding This research was partially supported by the Dutch research council (NWO) under grant no. VI.Veni.242.134 (VerHyp).

1 Introduction

Guarded Kleene Algebra with tests (GKAT) is a formalism for reasoning about control flow in simple imperative `while` programs. Its axiomatisation allows for equational reasoning, and it features an efficient automata-theoretic decision procedure for equivalence of programs [22]. GKAT has been extended to include restricted forms of non-local control flow [27], probabilities [17], weighted branching [8], and approximate reasoning [5].

GKAT programs are uninterpreted deterministic programs built from a base set of primitive programs and propositional tests. For instance, the if-then-else statement "if b holds then execute the uninterpreted program p, otherwise execute q", is expressed concisely as $p +_b q$. The operator $+_b$ is a guarded choice operator, closely related to branching operations common in process calculi [21], e.g., probabilistic choice [23]. GKAT is a fragment of *Kleene algebra with tests (KAT)* [9], likewise allowing algebraic reasoning about program equivalence. One source of motivation to study a fragment of KAT rather than KAT itself is the complexity of deciding program equivalence: decidability in KAT is PSPACE-complete, but program equivalence in GKAT can be decided in nearly linear time [22].

Like in KAT, GKAT programs are strictly propositional, so equivalence in GKAT is agnostic about the exact nature of each primitive program and test. This results in a notion of equivalence that only sees the control-flow structure of programs. In practice, however, one usually has access to further information about the (sub)programs at hand, and would like to incorporate such information when reasoning about program equivalence.


© Jurriaan Rot, Todd Schmid, and Jana Wagemaker;
licensed under Creative Commons License CC-BY 4.0

A motivating example for this paper arises from the wish to model notions of *hoisting* in programming languages, as expressed for instance by the following equation in a language with variable assignments [13]:

$$\texttt{if } b \texttt{ then } (x \leftarrow t);\ u \texttt{ else } (x \leftarrow t);\ v \quad = \quad (x \leftarrow t);\ \texttt{if } b \texttt{ then } u \texttt{ else } v \tag{1}$$

Above, u and v are arbitrary programs, $(x \leftarrow t)$ represents the assignment of a value t to the variable x, and b is a Boolean expression. An equation such as (1) typically holds when x does not occur free in the guard b. In GKAT, the above hoisting equation is modelled by abstracting away the details of the actions that occur. Specifically, the program action $(x \leftarrow t)$ is replaced by a formal *action symbol* p. Doing the same for u and v, if we let q represent u and r represent v, we can express the equivalence in (1) as follows in GKAT:

$$pq +_b pr = p(q +_b r) \tag{2}$$

For ordinary programs, in which the tests and constituent programs are interpreted, (2) is only valid when the execution of the program p has no effect on the truth value of b. For example, in (1) this lack of effect occurs when the variable x does not occur free in the guard b. We abstractly model this at the level of GKAT as follows: we assert the following two equations between programs.

$$bp = bpb \quad \text{and} \quad \overline{b}p = \overline{b}p\overline{b} \tag{3}$$

Here, a guard b appearing outside of a $+_b$ represents an assertion. The first equation says that if b holds before execution of p then it also holds afterwards, and the second says the same for the complement $\overline{b}$. Together, these equations assert that p does not affect the validity of b. Assuming (3), the hoisting equation (2) is valid in GKAT.

In this paper, we aim to support reasoning in GKAT under hypotheses such as (3). In the general form $bp = bpc$, where b and c are tests and p is a primitive program (what we will call an *action*), these equations are referred to as *Hoare hypotheses*. Hoare hypotheses are the propositional counterparts to *Hoare triples*: these are formulas written $\{P\}C\{Q\}$, where P and Q are interpreted tests and C is an interpreted program, and P is a sufficient starting condition to ensure Q holds after C is run [6]. Hoare hypotheses have already been considered in the context of Kleene algebra with tests [9, 12, 10], where decidability and completeness results are proven for the resulting equational theory. In fact, in plain Kleene algebra, the theory of reasoning about equivalence under such hypotheses is well developed, with both completeness and decidability results for classes of hypotheses that go well beyond Hoare hypotheses [2, 11, 4, 16].

However, in the context of GKAT, the addition of hypotheses remains entirely open. It is challenging even to define a suitable (sound) interpretation of programs in the presence of hypotheses, let alone decidability and completeness of the new theory. Without hypotheses, GKAT programs are interpreted in the same way as KAT programs, as languages of words whose letters alternate between test and action symbols. The general technique used in KA and KAT for adapting their semantics to account for hypotheses is to "blow up" the language by adding all words necessary to ensure soundness of the considered hypotheses [4, 16]. Subsequently, an expression is identified that describes the resulting language, and the problem of checking equivalence then reduces to the equivalence of the resulting expressions representing this semantics under hypotheses. This technique is not applicable to GKAT: GKAT programs are deterministic, so to be able to recover an expression that represents the semantics under hypotheses, the language has to remain deterministic. The novelty of

our approach to interpreting GKAT expressions in the presence of hypotheses is that it *cuts away* all irrelevant words. Determinism is what allows our decision procedure to retain the nearly-linear time complexity of ordinary GKAT [22], whereas a decision procedure based on KAT (with hypotheses) would likely be in PSPACE.

We take the first step towards developing a theory of *GKAT with hypotheses*, focusing on what we call *word hypotheses*. Word hypotheses are a generalization of the propositional Hoare hypotheses studied in the context of KAT in [10], and they state that for a particular alternation of programs and tests, a specific postcondition needs to hold. Propositional Hoare hypotheses are of the form $bp = bpc$, also discussed above in the example related to hoisting. Word hypotheses generalise this to equations of the form $b_0p_1b_1p_2 \dots b_{n-1}p_n = b_0p_1b_1p_2 \dots b_{n-1}p_nb_n$, where all b_i's are tests and p_i's are programs.

We start with the simpler form of Hoare hypotheses in Section 3. There, we introduce a language semantics, and prove completeness of our axiomatisation with respect to that semantics. We then move to the main contribution of this paper in Section 4: a decision procedure for general word hypotheses in nearly linear time. The key idea is to construct so-called *queue automata*, which remember the last read tests and programs, and which can detect the violation of a hypothesis. This yields precisely the behaviour of GKAT programs under word hypotheses, where program executions that violate a hypothesis are cut away. By showing that these queue automata reduce to plain GKAT automata, we obtain a reduction of decidability of equivalence to the original automata-theoretic decision procedure for equivalence of GKAT programs from [22], preserving its near linear time complexity. Basic notions and notation are covered in a preliminary section, Section 2.

Related work. For plain Kleene algebra, there is a well-developed tradition and theory of reasoning under additional hypotheses [11, 4, 2, 16], often focusing on completeness and decidability w.r.t. a model that is canonically defined from the given hypotheses. Hypothesis elimination in the context of Kleene algebra is also studied in relational models; see, e.g., [14]. Further, as mentioned above, Hoare hypotheses in the context of Kleene algebra with tests and associated reasoning was systematically studied in [10]. The current paper instead focuses specifically on the addition of hypotheses to GKAT and its model of guarded strings.

Propositional Hoare triples are considered in [22], in the context of a completeness theorem for GKAT. However, only the Hoare triples that are already valid in GKAT are considered, and no reasoning under new hypotheses is studied. The Hoare triples studied in this paper are typically *not* valid in GKAT, but are additional assumptions made about specific programs.

Program logics like NetKAT [1] and Guarded NetKAT [25] involve reasoning about programs in a similar fashion to KAT and GKAT, but with additional hypotheses formally imposed by insight into the nature of primitive actions and tests. The desire to capture the structure of these program logics abstractly provides further motivation for the current work.

2 Guarded Kleene Algebra with Tests

Guarded Kleene Algebra with Tests (*GKAT*) is an abstraction from standard imperative programming: its basic elements are a collection of *primitive programs* Σ and a finite collection of *primitive tests* T, and all other programs expressible in the language of GKAT are formed by combining these elements with `if-then-else` and `while-do` constructs. Formally, the collection *GExp* of *GKAT expressions* is given by the BNF grammar

$$GExp \ni e, f ::= b \in BExp \mid p \in \Sigma \mid e +_b f \mid e \cdot f \mid e^{(b)}$$
$$BExp \ni b, c ::= t \in T \mid \bar{b} \mid b \vee c$$

Intuitively understood, each test $b \in BExp$ (thought of as a GKAT expression) is an assertion $\texttt{assert}\ b$, the expression $e +_b f$ is conditional branching $\texttt{if}\ b\ \texttt{then}\ e\ \texttt{else}\ f$, $e \cdot f$ is sequential composition, and $e^{(b)}$ is the conditional loop $\texttt{while}\ b\ \texttt{do}\ e$. For brevity, we use juxtaposition ef instead of $\cdot$ when the meaning is clear. We define $b \wedge c = \overline{(\bar{b} \vee \bar{c})}$, $1 = b \vee \bar{b}$, and $0 = b \wedge \bar{b}$. We also sometimes write $+_{bc}$ instead of $+_{b \wedge c}$.

Since we have assumed that T is finite, there are only finitely many tests modulo the Boolean algebra axioms, and this in particular implies that there is a well defined notion of *atomic test*. Specifically, an *atomic test* is a Boolean expression of the form $\sigma(t_1) \wedge \cdots \wedge \sigma(t_n)$, where $T = \{t_1, \ldots, t_n\}$ and $\sigma(t_i) \in \{t_i, \overline{t_i}\}$ for each $i \leq n$. The set of atomic tests in $BExp$ is written At, and we use the Greek letters α, β for atomic tests. Thinking of tests as descriptions of program states (like how the test $\texttt{x} > 0$ describes program states where the variable $\texttt{x}$ takes a value above 0), an atomic test represents a complete description of one specific program state, insofar as the status of the tests of the program affects its flow. It is worth noting that the number of atomic tests is exponential in the number of primitive tests.

Language semantics. GKAT expressions are abstract representations of programs that have a well defined notion of program trace. Just like in KAT [9], these traces take the form of *guarded strings*, which are alternating sequences of atomic tests and primitive programs. For a set X we denote by X^* the set of all words over X, with the empty word denoted by ε.

▶ **Definition 1.** *A* guarded string *is a word $w = \alpha_0 p_1 \alpha_1 \cdots \alpha_{n-1} p_n \alpha_n$ whose letters alternate between atomic tests and primitive programs, i.e., $\alpha_0, \ldots, \alpha_n \in At$ and $p_1, \ldots, p_n \in \Sigma$. Formally, the set of guarded strings is $(At \cdot \Sigma)^* \cdot At$. A* language *of guarded strings, or* guarded language*, is a set of guarded strings. We write GL for the set of guarded languages $GL = \mathcal{P}((At \cdot \Sigma)^* \cdot At)$.*

Equating program traces with guarded strings leads to the standard interpretation of GKAT expressions, the *language semantics*. To state the language semantics precisely, we need to define the product of two guarded languages $L_1, L_2 \in GL$. To this end, we define the *product* $\diamond$ of two guarded strings $u\alpha, \beta v \in (At \cdot \Sigma)^* \cdot At$ to be $u\alpha v$ if $\alpha = \beta$ and undefined otherwise. This is then lifted to languages:

$$L_1 \diamond L_2 = \{u\alpha \diamond \alpha v \mid u\alpha \in L_1,\ \alpha v \in L_2\} \tag{4}$$

Moreover, we define an associated Kleene star of a guarded language L by $L^\star = \bigcup_{n \geq 0} L^{\langle n \rangle}$ where $L^{\langle 0 \rangle} = At$ and $L^{\langle n+1 \rangle} = L \diamond L^{\langle n \rangle}$.

We write BA for the equational theory of Boolean algebras, $=_{\mathsf{BA}}$ for the congruence generated by BA, and $\leq_{\mathsf{BA}}$ for the order induced by BA, i.e., $b \leq_{\mathsf{BA}} c$ *iff* $b \vee c =_{\mathsf{BA}} c$. For ease of notation, we also define $b \diamond L = \{\alpha \in At \mid \alpha \leq_{\mathsf{BA}} b\} \diamond L$ and $L \diamond b = L \diamond \{\alpha \in At \mid \alpha \leq_{\mathsf{BA}} b\}$. Later, we will make use of the fact that $(b \diamond L) \cap K = b \diamond (L \cap K)$ and $(L \diamond b) \cap K = (L \cap K) \diamond b$ for any guarded languages L, K and $b \in BExp$.

▶ **Definition 2.** *The* language interpretation *(or* language semantics*) $\mathcal{L}(e)$ of a GKAT expression $e \in GExp$ is defined recursively on $e \in GExp$ by*

$$\mathcal{L}(b) = \{\alpha \in At \mid \alpha \leq_{\mathsf{BA}} b\} \quad \mathcal{L}(p) = \{\alpha p \beta \mid \alpha, \beta \in At\}$$
$$\mathcal{L}(e +_b f) = b \diamond \mathcal{L}(e) \cup \bar{b} \diamond \mathcal{L}(f) \quad \mathcal{L}(ef) = \mathcal{L}(e) \diamond \mathcal{L}(f) \quad \mathcal{L}(e^{(b)}) = (b \diamond \mathcal{L}(e))^\star \diamond \bar{b}$$

Thus, we obtain the semantic map $\mathcal{L}\colon GExp \to GL$. *Given $e, f \in GExp$, if $\mathcal{L}(e) = \mathcal{L}(f)$, we say that e and f are* language equivalent.

The following theorem is [22, Theorem 5.10], which states the decidability and time complexity of deciding language equivalence of GKAT expressions, assuming that the sets of atomic tests At and alphabet Σ are held constant. Given an expression $e \in GExp$, we write $|e|$ for the number of symbols that appear in e.

▶ **Theorem 3** (Decidability in Nearly-Linear Time [22]). *Let $e, f \in GExp$. For $|At|$ held constant, it is decidable in $\mathcal{O}(n\alpha(n))$-time whether $\mathcal{L}(e) = \mathcal{L}(f)$, where $\alpha(n)$ is the inverse Ackermann function and $n = |e| + |f|$.*

The complexity class $\mathcal{O}(n\alpha(n))$ is called *nearly-linear* because of the extremely slow growth of the inverse Ackermann function $\alpha(n)$. The algorithm in [22] and its complexity analysis relies on [7, 24], and requires explicit calculation of At and Σ ahead of time. Without the assumptions that $|At|$ and Σ are held fixed, the automata generated by GKAT expressions incur an exponential blow-up (because $|At| = 2^{|T|}$). This is a known limitation of existing decision procedures, and is implicit even in earlier work on KAT [3]. Recently developed symbolic methods (going back to [15]) appear to avoid this issue in practice; see [26].

Axiomatisation. One of the key features of GKAT is that it allows for equational reasoning between programs. In [22], a sound and complete axiomatisation of language equivalence for GKAT expressions is presented. We refer to that axiomatisation as GKAT, and record the axioms (other than the Boolean algebra axioms BA and equational logic) in Figure 1. We write $\mathsf{GKAT} \vdash e = f$ if the equality $e = f$ is derivable from the axioms in GKAT.

Axioms (W3) and (UA) in Figure 1 use the map $E\colon GExp \to BExp$ defined recursively by

$$E(b) = b \qquad E(p) = 0 \qquad E(e +_b f) = bE(e) \vee \bar{b}E(f)$$
$$E(ef) = E(e) \wedge E(f) \qquad E(e^{(b)}) = \bar{b}$$

for any $e \in GExp$. Intuitively, $E(e)$ is the least condition that guarantees e immediately halts and accepts, so $E(e) =_{\mathsf{BA}} 0$ means that the program e performs an action under any circumstance. In terms of languages, $E(e) = \bigvee(At \cap \mathcal{L}(e))$. The side condition $E(e) =_{\mathsf{BA}} 0$ is analogous to the side condition regarding the *empty word property* in Salomaa's axioms for the algebra of regular languages [18]. These side conditions are necessary to ensure soundness of (W3) and (AU): for example, by (W1), $\mathsf{GKAT} \vdash 1 = 1 \cdot 1 +_1 1$, but $\mathcal{L}(1) \neq \mathcal{L}(1^{(1)}) = \emptyset$ [22].

▶ **Theorem 4** ([22]). *For any $e, f \in GExp$, $\mathsf{GKAT} \vdash e = f$ iff $\mathcal{L}(e) = \mathcal{L}(f)$.*

Note the presence of the axiom (UA). This axiom is an extenson of (W3) to arbitrary systems of equations, ensuring that solutions to systems (satisfying the side condition) are unique when they exist. It is still open whether (UA) is derivable from the other axioms [20].

3 Hoare hypotheses

In this section, we incorporate *Hoare hypotheses* into GKAT (see Definition 6). We introduce their semantics, prove completeness, and obtain decidability as a consequence of the underlying construction of the completeness proof. An analysis of the complexity of this decision procedure appears at the end of the section, but it is worth noting in advance that it is not as efficient as the decision procedure we introduce later in Section 4 for word hypotheses.

Let us begin with the general definition of *hypothesis* in the context of GKAT.

▶ **Definition 5.** *A* hypothesis *is a formal equation $e = f$, where $e, f \in GExp$. For a set H of hypotheses, we write $\mathsf{GKAT} + H \vdash e = f$ when $e = f$ is derivable from the axioms in* GKAT *and the hypotheses in H.*

Union Axioms

(U1) $e +_b e = e$

(U2) $e +_b f = f +_{\bar{b}} e$

(U3) $(e +_b f) +_c g = e +_{bc} (f +_c g)$

(U4) $e +_b f = be +_b f$

(U5) $eg +_b fg = (e +_b f)g$

Sequencing Axioms

(S1) $(ef)g = e(fg)$

(S2) $0e = 0$

(S3) $e0 = 0$

(S4) $1e = e$

(S5) $e1 = e$

(S6) $bc = b \wedge c$

Loop Axioms

(W1) $e^{(b)} = ee^{(b)} +_b 1$

(W2) $(e +_c 1)^{(b)} = (ce)^{(b)}$

(W3) $$\frac{g = eg +_b f \quad E(e) =_{\mathsf{BA}} 0}{g = e^{(b)} f}$$

(UA) $$\frac{\{g_i = e_{1i}g_1 +_{b_{1i}} \cdots e_{ni}g_n +_{b_{ni}} d_i\}_{i \le n} \quad \{f_i = e_{1i}f_1 +_{b_{1i}} \cdots e_{ni}f_n +_{b_{ni}} d_i\}_{i \le n} \quad \{b_{ij}b_{ik} =_{\mathsf{BA}} 0\}_{j \neq k, i \le n} \quad \{E(e_{ij}) =_{\mathsf{BA}} 0\}_{i,j \le n}}{g_i = f_i}$$

Figure 1 [22] Axioms for proving program equivalence of GKAT-expressions. In the GKAT expressions that appear above, $e, f, g, e_{ij}, f_i, g_i \in \mathit{GExp}$ and $b, c, b_{ij}, d_i \in \mathit{BExp}$.

Intuitively, hypotheses extend our knowledge of the meanings of GKAT programs by imposing additional equations. Following this intuition, *Hoare hypotheses*, formally introduced below, can be thought of as statements of the form, "if the test b holds before executing the primitive program p, then the test c holds after p executes".

▶ **Definition 6** (Hoare hypothesis). *A* Hoare hypothesis *is a hypothesis of the form $bp = bpc$, where $b, c \in \mathit{BExp}$ and $p \in \Sigma$. If H is a set of Hoare hypotheses, then we say that* $\mathsf{GKAT} + H$ *is a* Hoare extension *of* GKAT*.*

Simple as they are, Hoare hypotheses in GKAT are able to capture a number of different ways in which imperative programs might interact. To illustrate the expressiveness of Hoare hypotheses in GKAT, let us return to our motivating application: a phenomenon in imperative programming called *hoisting*, which allows one to "hoist" a program that appears at the start of each branch of an `if − then − else` construct out to just before the conditional, as in (1). Given a program p and a test b, the *hoisting principle for p over b* is the set of hypotheses

$$\mathsf{Hst}_{p,b} = \{p(e +_b f) = pe +_b pf \mid e, f \in \mathit{GExp}\}$$

As discussed in the introduction, these hoisting principles are not already valid in GKAT. However, they become valid with the addition of two Hoare hypotheses that state the preservation of the test b by the execution of the primitive program p. Formally, the *invariant test hypothesis for b under p* is the set of Hoare hypotheses

$$\mathsf{Inv}_{p,b} = \{bp = bpb, \bar{b}p = \bar{b}p\bar{b}\}$$

In fact, invariance is equivalent to hoisting.

▶ **Lemma 7.** *Let $e, f \in \mathit{GExp}$.* $\mathsf{GKAT} + \mathsf{Hst}_{p,b} \vdash e = f$ *if and only if* $\mathsf{GKAT} + \mathsf{Inv}_{p,b} \vdash e = f$.

Each hypothesis $bp = bpb$ is essentially the statement that the validity of b is "maintained" through the execution of p. Intuitively, this could also be expressed as the statement that running p before the assertion `assert` b has the same effect as running it after; i.e., that b and p *commute*. The *commutative principle for* b *and* p is the set of two hypotheses

$$\mathsf{Com}_{p,b} = \{pb = bp, p\bar{b} = \bar{b}p\}$$

It turns out that these are also equivalent to invariant test hypotheses.

▶ **Lemma 8.** *Let* $e, f \in GExp$. $\mathsf{GKAT} + \mathsf{Inv}_{p,b} \vdash e = f$ *if and only if* $\mathsf{GKAT} + \mathsf{Com}_{p,b} \vdash e = f$.

Semantics and Completeness. Given a Hoare extension $\mathsf{GKAT} + H$ of GKAT, we provide a semantics $\mathcal{L}_H(-)$ such that $\mathsf{GKAT} + H \vdash e = f$ iff $\mathcal{L}_H(e) = \mathcal{L}_H(f)$. The idea behind our semantics for GKAT with Hoare hypotheses is to adjust the ordinary language semantics of GKAT by restricting it to the guarded strings that satisfy the hypotheses. Later, we will show how to reduce semantic equivalence to ordinary language equivalence of GKAT and appeal to Theorem 3 for decidability.

Intuitively, the language semantics of a GKAT expression e (in general) consists of all program traces that are possible during an execution of e. The basic idea, then, is that under a hypothesis $bp = bpc$, there are fewer of these possible program traces. In particular, if a trace contains a step $\alpha_i p$ with $\alpha_i \leq_{\mathsf{BA}} b$, then the execution of p must proceed with an α_{i+1} in which c holds. Our semantics therefore removes from the ordinary language semantics all traces that do not satisfy this condition.

▶ **Definition 9.** *Let* H *be a set of Hoare hypotheses. Given* $e \in GExp$*, we define the* language semantics up to H *of* e *to be the guarded language*

$$\mathcal{L}_H(e) = \mathcal{L}(e) \cap \left\{ \alpha_0 p_1 \cdots \alpha_{n-1} p_n \alpha_n \;\middle|\; \begin{array}{l} \text{for all } i \leq n, \text{if } bp_i = bp_i c \in H \\ \text{and } \alpha_i \leq_{\mathsf{BA}} b, \text{ then } \alpha_{i+1} \leq_{\mathsf{BA}} c \end{array} \right\}$$

Given $e, f \in GExp$*, we say that* e *and* f *are* language equivalent up to H *if* $\mathcal{L}_H(e) = \mathcal{L}_H(f)$.

It will be useful to observe that $\mathcal{L}_H$ is a homomorphism, in the following sense.

▶ **Lemma 10.** *For any set of Hoare hypotheses* H*,* $e, f \in GExp$ *and* $b \in BExp$*:*

$$\mathcal{L}_H(b) = \{\alpha \in At \mid \alpha \leq_{\mathsf{BA}} b\} \qquad \mathcal{L}_H(ef) = \mathcal{L}_H(e) \diamond \mathcal{L}_H(f)$$
$$\mathcal{L}_H(e +_b f) = b \diamond \mathcal{L}_H(e) \cup \bar{b} \diamond \mathcal{L}_H(f) \qquad \mathcal{L}_H(e^{(b)}) = (b \diamond \mathcal{L}_H(e))^\star \diamond \bar{b}$$

The axioms of GKAT with Hoare hypotheses are sound with respect this semantics.

▶ **Theorem 11** (Soundness)**.** *Let* H *be a set of Hoare hypotheses. For any* $e, f \in GExp$*, if* $\mathsf{GKAT} + H \vdash e = f$*, then* $\mathcal{L}_H(e) = \mathcal{L}_H(f)$.

To prove completeness, we adapt a standard approach from Kleene algebra with hypotheses (e.g., [16]) to the GKAT setting: we construct a *reduction*. That is, we define a syntactic translation of GKAT expressions that incorporates the hypotheses into any given expression. We start by defining the reduction for a single hypothesis.

▶ **Definition 12.** *Let* h *be the Hoare hypothesis* $bp = bpc$*. The* translation *function* $T_h \colon GExp \to GExp$ *is the homomorphic extension of the map defined by*

$$T_h(d) = d \qquad T_h(p) = pc +_b p \qquad T_h(q) = q$$

where $d \in BExp$ *and* $q \in \Sigma$ *with* $p \neq q$.

Given a hypothesis h, the translation $T_h(e)$ of an expression e is provably equivalent to e up to the axioms of GKAT and the hypothesis h.

▶ **Lemma 13.** *For any $e \in GExp$, $\mathsf{GKAT} + h \vdash T_h(e) = e$.*

Proof. By induction on e. The inductive cases are covered because T_h is a homomorphism. The only interesting base case is where $e = p$ and where the hypothesis h is $bp = bpc$. We have the following derivation, in which we use (U4), (h), (U4), and (U1), in that order:

$$\mathsf{GKAT} + h \vdash T_h(p) = pc +_b p = bpc +_b p = bp +_b p = p +_b p = p$$ ◀

We are now in a position to define the reduction for multiple Hoare hypotheses.

▶ **Definition 14.** *Let H be a set of Hoare hypotheses. The translation $T_H : GExp \to GExp$ of $e \in GExp$ is defined inductively by*

$$T_H(e) = T_h(T_{H \setminus h}(e))$$

where $H \setminus h$ is the set of Hoare hypotheses H excluding h.

This is well defined because of the following lemma, which shows that the order in which the translations for individual Hoare hypotheses in H are applied is immaterial.

▶ **Lemma 15.** *For any Hoare hypotheses h_1 and h_2, the translations commute:*

$$\mathsf{GKAT} + H \vdash T_{h_1} T_{h_2}(e) = T_{h_2} T_{h_1}(e).$$

Proof. Suppose h_i is the hypothesis $b_i p_i = b_i p_i c_i$ for $i = 1, 2$. We proceed by induction on e. The only interesting case is when $e = p_1 = p_2$, since the other cases are either trivial applications of the induction hypothesis or the two translations do not interact. So, let $p = p_1 = p_2$. In the calculation below, we will use the identity $\mathsf{GKAT} \vdash e +_b (f +_c g) = (e +_b f) +_{b \vee c} g$, which we denote (U3'). This is also (U3') in [22], where it is proven by repeatedly using (U2) and (U3) and the Boolean algebra axioms. We have

$$\begin{aligned}
\mathsf{GKAT} + H \vdash T_{h_1} T_{h_2}(p) &= T_{h_1}(pc_2 +_{b_2} p) && \text{(def. } T_{h_2}\text{)} \\
&= (pc_1 +_{b_1} p)c_2 +_{b_2} (pc_1 +_{b_1} p) && \text{(def. } T_{h_1}\text{)} \\
&= (pc_1c_2 +_{b_1} pc_2) +_{b_2} (pc_1 +_{b_1} p) && \text{(U5)} \\
&= pc_1c_2 +_{b_1 b_2} (pc_2 +_{b_2} (pc_1 +_{b_1} p)) && \text{(U3)} \\
&= pc_1c_2 +_{b_1 b_2} ((pc_2 +_{b_2} pc_1) +_{b_1 \vee b_2} p) && \text{(U3')} \\
&= pc_1c_2 +_{b_1 b_2} ((pc_1 +_{\bar{b}_2} pc_2) +_{b_1 \vee b_2} p) && \text{(U2)} \\
&= pc_1c_2 +_{b_1 b_2} (pc_1 +_{b_1} (pc_2 +_{b_1 \vee b_2} p)) && \text{(U3, BA)} \\
&= pc_1c_2 +_{b_1 b_2} ((pc_1 +_{b_1} pc_2) +_{b_1 \vee b_2} p) && \text{(U3', BA)} \\
&\quad \vdots && \text{(same steps backwards)} \\
&= T_{h_2} T_{h_1}(p) &&
\end{aligned}$$ ◀

A routine induction on the number of hypotheses allows us to extend Lemma 13 to an arbitrary finite set of Hoare hypotheses H.

▶ **Theorem 16.** *For any $e \in GExp$, $\mathsf{GKAT} + H \vdash T_H(e) = e$.*

The following relationship between the language model of GKAT and the language model $\mathcal{L}_H$ of $\mathsf{GKAT} + H$ is core to the proof that T_H is a meaningful translation of GKAT expressions.

▶ **Theorem 17.** *Let $e \in GExp$. Then $\mathcal{L}_H(e) = \mathcal{L}(T_H(e))$.*

Proof. We proceed by induction on e. If $e = b \in BExp$, the result trivially holds because $T_H(b) = b$ and $\mathcal{L}_H(b) = \mathcal{L}(b)$ by definition.

For $e = p \in \Sigma$, we proceed by induction on the number of hypotheses in H that involve p. If there are no hypotheses in H that involve p, the result trivially holds. Otherwise, pick any $h \in H$ of the form $bp = bpc$ for some $b, c \in BExp$. As the order of translations in T_H does not matter (Lemma 15), we assume the translation T_h is performed "last", i.e. $T_H(e) = T_h(T_{H\setminus h}(e))$. Our induction hypothesis states that $\mathcal{L}_{H\setminus h}(p) = \mathcal{L}(T_{H\setminus h}(p))$. We prove $\mathcal{L}_H(p) = \mathcal{L}(T_H(p))$, as follows:

$$
\begin{aligned}
\mathcal{L}_H(p) &= \mathcal{L}_H(bp +_b \bar{b}p) && \text{(basic property of } \mathcal{L}\text{, therefore also } \mathcal{L}_H) \\
&= (b \diamond \mathcal{L}_H(p)) \cup (\bar{b} \diamond \mathcal{L}_H(p)) && \text{(Lemma 10)} \\
&= (b \diamond \mathcal{L}_{H\setminus h}(p) \diamond c) \cup (\bar{b} \diamond \mathcal{L}_{H\setminus h}(p)) && \text{(from def. } \mathcal{L}_H, \mathcal{L}_{H\setminus h}) \\
&= (b \diamond \mathcal{L}(T_{H\setminus h}(p)) \diamond c) \cup (\bar{b} \diamond \mathcal{L}(T_{H\setminus h}(p))) && \text{(Induction Hypothesis)} \\
&= (b \diamond \mathcal{L}(T_{H\setminus h}(p)c)) \cup (\bar{b} \diamond \mathcal{L}(T_{H\setminus h}(p))) && \text{(def. } \mathcal{L}) \\
&= \mathcal{L}(T_{H\setminus h}(p)c +_b T_{H\setminus h}(p)) && \text{(def. } \mathcal{L}) \\
&= \mathcal{L}(T_{H\setminus h}(pc +_b p)) && \text{(def. } T_{H\setminus h}) \\
&= \mathcal{L}(T_{H\setminus h}(T_h(p))) && \text{(def. } T_h) \\
&= \mathcal{L}(T_h(T_{H\setminus h}(p))) && \text{(Lemma 15)} \\
&= \mathcal{L}(T_H(p))
\end{aligned}
$$

This concludes the case $e = p \in \Sigma$.

For the induction step $e +_b f$, we get:

$$
\begin{aligned}
\mathcal{L}_H(e +_b f) &= b \diamond \mathcal{L}_H(e) \cup \bar{b} \diamond \mathcal{L}_H(f) && \text{(Lemma 10)} \\
&= b \diamond \mathcal{L}(T_H(e)) \cup \bar{b} \diamond \mathcal{L}(T_H(f)) && \text{(Induction Hypotheses)} \\
&= \mathcal{L}(T_H(e) +_b T_H(f)) && \text{(def. } \mathcal{L}) \\
&= \mathcal{L}(T_H(e +_b f)) && \text{(def. } T_H)
\end{aligned}
$$

The other inductive cases $e \diamond f$ and $e^{(b)}$ can be proven similarly from Lemma 10. ◀

Theorems 16 and 17 lead us to our completeness and decidability theorems. Together, they allows us to reduce equivalence of $\mathsf{GKAT} + H$ to GKAT equivalence.

▶ **Theorem 18** (Completeness). *Let $e, f \in GExp$. If $\mathcal{L}_H(e) = \mathcal{L}_H(f)$, then $\mathsf{GKAT} + H \vdash e = f$.*

Proof. Let $e, f \in GExp$. Using Theorem 17, completeness of GKAT, and Theorem 16,

$$
\begin{aligned}
\mathcal{L}_H(e) = \mathcal{L}_H(f) &\Leftrightarrow \mathcal{L}(T_H(e)) = \mathcal{L}(T_H(f)) \\
&\Rightarrow \mathsf{GKAT} \vdash T_H(e) = T_H(f) \\
&\Leftrightarrow \mathsf{GKAT} + H \vdash e = f.
\end{aligned}
$$

◀

▶ **Theorem 19** (Decidability). *Let $e, f \in GExp$. It is decidable whether $\mathcal{L}_H(e) = \mathcal{L}_H(f)$.*

Proof. To decide whether $\mathcal{L}_H(e) = \mathcal{L}_H(f)$, via Theorem 17 it suffices to decide whether $\mathcal{L}(T_H(e)) = \mathcal{L}(T_H(f))$. This can be done using Theorem 3 and that T_H is computable. ◀

Let us take a moment to remark on the complexity of the decision procedure suggested by the completeness proof above. The decision procedure reduces the problem to deciding ordinary language equivalence of two GKAT expressions e, f. We know from Theorem 3 that equivalence is decidable in $\mathcal{O}(n\alpha(n))$ time, where $n = |e| + |f|$. In the presence of a set of Hoare hypotheses H, the decision procedure described in our completeness proof would therefore run in $\mathcal{O}(m\alpha(m))$ time, where $m = |T_H(e)| + |T_H(f)|$. To measure the complexity of this decision procedure, we need only to compute the sizes of $T_H(e)$ and $T_H(f)$. We write $p \in e$ to express that $p \in \Sigma$ occurs in e, and let n_p^H be the number of hypotheses in H that contain a given action p.

▶ **Lemma 20.** *Let $e \in GExp$ and let H be a set of Hoare hypotheses. Then*

$$|T_H(e)| = \sum_{p \in e} (3(2^{n_p^H}) - 3) + |e|$$

It follows that the decision procedure suggested by the completeness proof runs in exponential time in the worst case. Fortunately, a much more efficient decision procedure exists that applies to a slightly broader class of hypotheses, although the same completeness proof technique does not appear to apply. The larger class of hypotheses and their corresponding decision procedure is the subject of the next section.

4 Decidability for Word Hypotheses

In this section we consider a generalisation of Hoare hypotheses that we call *word hypotheses*. We present a decision procedure for program equivalence under word hypotheses that maintains the nearly linear time complexity of GKAT. The decision procedure is based on a construction that adds a queue-like data structure (called an *inspection queue*) to *GKAT automata*, which are the operational models of GKAT (called *G-coalgebras* in [22]).

As we will see, Hoare hypotheses are all instances of word hypotheses, so the presented decision procedure is a more efficient decision procedure for Hoare hypotheses than the one suggested by Theorem 19. Essentially, the automata-based decision procedure avoids computing the reduction map T_H, which led to an exponential blow-up.

Word hypotheses. Recall that a guarded string is an element of $(At \cdot \Sigma)^* \cdot At$. We call an element of $(At \cdot \Sigma)^*$ a *pre-guarded string*, because each of these is a prefix of a guarded string. Roughly, a *word hypothesis* is a constraint on the guarded strings admissible in a language that begin with a given pre-guarded string.

▶ **Definition 21.** *A* word hypothesis *is a formal identity of the form $w = wc$ for some alternating concatenation of tests and atomic programs $w \in (BExp \cdot \Sigma)^*$ and some test $c \in BExp$. A guarded string $\alpha_0 p_1 \alpha_1 \cdots \alpha_{l-1} p_l \alpha_l$* violates *$w = wc$ if $w = b_0 p_1 b_1 \cdots b_{l-1} p_l$, $\alpha_i \leq_{\mathsf{BA}} b_i$ for all $i < l$, and $\alpha_l \leq_{\mathsf{BA}} \bar{c}$. A guarded string* violates *a set of word hypotheses H if it violates a word hypothesis in H.*

▶ **Example 22.** In programming terms, a word hypothesis states a certain relationship between runs of a program and a post-condition. For a simple concrete example, consider the following alternating sequence of tests and programs involving integer variables $\mathtt{x}$ and $\mathtt{y}$:

$$w = (\mathtt{y} > 1)(\mathtt{x} \leftarrow \mathtt{x} * \mathtt{y})(\mathtt{x} > 0)(\mathtt{x} \leftarrow \mathtt{x} * \mathtt{y}) \tag{5}$$

The bracketed expressions involving the $>$ symbol are tests and the expressions involving $\leftarrow$ are assignments, so w can appear on the left-hand-side of a word hypothesis. Since $\mathtt{x}$ and $\mathtt{y}$ are integer variables, $(\mathtt{x} > 0)$ and $(\mathtt{y} > 1)$ imply that the values of both are at least 1 and 2

respectively. One word hypothesis that would be reasonable in this example is $w = w(\mathtt{x} > 4)$, since running w multiplies $\mathtt{x}$ (which is at least 1) by $\mathtt{y}$ (which is at least 2) twice. Given any atomic test $\alpha \in At$ that implies both $(\mathtt{x} == 1)$ and $(\mathtt{y} == 2)$, the guarded string

$$\alpha(\mathtt{x} \leftarrow \mathtt{x} * \mathtt{y})\alpha(\mathtt{x} \leftarrow \mathtt{x} * \mathtt{y})\alpha$$

violates $w = w(\mathtt{x} > 4)$, because α passes the tests $(x > 0)$ and $(y > 1)$, but it fails $(x > 4)$.

Given a guarded string $w = \alpha_0 p_1 \cdots p_n \alpha_n$, we call a consecutive string of characters $\alpha_i p_{i+1} \cdots p_j \alpha_j$, for $0 \leq i < j \leq n$, a *guarded substring* of w.

▶ **Definition 23.** *Let H be a set of word hypotheses. Given $e \in GExp$, we define*

$$\mathcal{L}_H(e) = \mathcal{L}(e) \cap \{w \mid \textit{no guarded substring of } w \textit{ violates } H\}$$

The set of guarded strings $\mathcal{L}_H(e)$ is the language semantics up to H *of e. For any $e, f \in GExp$, if $\mathcal{L}_H(e) = \mathcal{L}_H(f)$, then we say that e and f are* language equivalent up to H*.*

Let us define the *length* $\mathsf{len}(w)$ of a (pre)guarded string w to be the number of primitive programs that appear in it. That is, $\mathsf{len}(\alpha_0 p_1 \cdots p_n \alpha_n) = n = \mathsf{len}(\alpha_0 p_1 \cdots p_n)$. We take the length $\mathsf{len}(w = wc)$ of a word hypothesis $w = wc$ to be the length of w. In this terminology, Hoare hypotheses are word hypotheses of length 1, and the semantics of GKAT expressions up to word hypotheses is a direct generalisation of the semantics of GKAT expressions up to Hoare hypotheses from Definition 9. This justifies our reuse of the notation $\mathcal{L}_H$ for both notions of language semantics. Our next (and final) task of the paper is to show that language equivalence up to H is efficiently decidable.

Inspection Queues. The core of our automata-based decision procedure is the notion of a *bounded inspection queue*, which is a data structure built on the set of nonempty pre-guarded strings of some bounded length. Intuitively, a bounded inspection queue is a queue with bounded capacity that evicts the head of the queue when an item is enqueued and the queue is at maximum capacity. To state the definition precisely, we write $(At \cdot \Sigma)^{\leq l}$ for the set of pre-guarded strings of length at most l and define the following two functions.

- The *inspection function* is the map $\mathsf{insp} \colon (\mathbb{N} \cup \{\infty\}) \times (At \cdot \Sigma)^* \to (At \cdot \Sigma)^*$ defined by

$$\mathsf{insp}(n, w) = \begin{cases} v & w = uv \text{ and } \mathsf{len}(v) = n \\ w & \mathsf{len}(w) < n \end{cases}$$

  for any $n \in \mathbb{N} \cup \{\infty\}$ and pre-guarded string w. Intuitively, $\mathsf{insp}(n, -)$ returns the last n atomic test-action pairs in the input string.
- For any given *queue length* $l \in \mathbb{N} \cup \{\infty\}$, the *$l$-bounded enqueue function* is the map $\mathsf{enq}_l \colon (At \cdot \Sigma) \times (At \cdot \Sigma)^* \to (At \cdot \Sigma)^{\leq l}$ defined by $\mathsf{enq}_l(\alpha p, w) = \mathsf{insp}(l, w\alpha p)$. Intuitively, enq_l is the enqueueing map for a *bounded queue* of height l.

For an easy example, $\mathsf{insp}(\infty, w) = w$ always. For a less trivial example, in the process of 2-bounded enqueueing βq to the pre-guarded string $\alpha_0 p_1 \alpha_1 p_2$ (which is of length 2), the $\alpha_0 p_1$ portion of the queue is deleted, and the state of the queue becomes the string $\alpha_1 p_2 \beta q$.

$$\mathsf{enq}_2(\beta q, \alpha_0 p_1 \alpha_1 p_2) = \mathsf{insp}(2, \alpha_0 p_1 \alpha_1 p_2 \beta q) = \alpha_1 p_2 \beta q.$$

Our operational semantics for GKAT with word hypotheses essentially attaches a bounded queue to the standard operational model of GKAT expressions.

▶ **Definition 24** ([22]). *A* GKAT-automaton *is a pair (X, δ) consisting of a set X of* states *and a* transition function $\delta\colon At \times X \to 2 + (\Sigma \times X)$. *Here, $2 = \{\bot, \top\}$, and $\bot$ and $\top$ are called* reject *and* accept *respectively.*

Given $\alpha \in At$, $p \in \Sigma$, and a GKAT-automaton (X, δ) with states $x, y \in X$, we write $x \xrightarrow{\alpha|p} y$ if $\delta(\alpha, x) = (p, y)$, $x \Rightarrow \alpha$ if $\delta(\alpha, x) = \top$, and $x \downarrow \alpha$ if $\delta(\alpha, x) = \bot$. In diagrams, we will not include depictions of $x \downarrow \alpha$ transitions.

A guarded string $\alpha_0 p_1 \cdots p_n \alpha_n$ is accepted *by a state $x \in X$ if there is a path*

$$x \xrightarrow{\alpha_0|p_1} x_1 \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} x_n \Rightarrow \alpha_n,$$

that is, there is a path starting from x that reads $\alpha_0 p_1 \cdots p_n$ and then arrives at a state that accepts the atomic test α_n. The language accepted by *x is given by*

$$\mathcal{L}((X, \delta), x) = \{w \in (At \cdot \Sigma)^* \cdot At \mid x \text{ accepts } w\}.$$

Given a set of word hypotheses H, we use a queue to store a pre-guarded string w which represents (part of) the sequence of atomic tests and primitive programs that an automaton is reading as it runs. In particular, the automaton should "get stuck" if the state of the queue violates a word hypothesis in H.

▶ **Definition 25.** *Let H be a set of word hypotheses. Given $\alpha \in At$ and $w \in (At \cdot \Sigma)^*$, we say that α* is locked at *w if there is an $n \in \mathbb{N}$ s.t. the guarded string $\mathsf{insp}(n, w)\alpha$ violates H. That is, α is locked at w if there is a suffix u of w s.t. $u\alpha$ violates some word hypothesis in H.*

We now define the operational model of GKAT with word hypotheses. The idea is to take a set of word hypotheses H and a GKAT automaton (X, δ) and "explode" (X, δ) into the set of all (*program-state*, *queue-state*) pairs. This allows us to force a deadlock whenever a hypothesis is violated; more precisely, the transition function only retains those transitions for which the associated atomic test is not locked at the current queue state.

▶ **Definition 26.** *Let H be a set of word hypotheses and fix the queue length $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$ to be the maximum length of a word hypothesis in H (possibly, $l = \infty$, which occurs when H is infinite). Given a GKAT automaton (X, δ), we define the* inspection queue automaton *by $\mathcal{Q}_H(X, \delta) = ((At \cdot \Sigma)^{\leq l} \times X, \delta_H)$, where the transition function $\delta_H\colon At \times ((At \cdot \Sigma)^{\leq l} \times X) \to 2 + \Sigma \times ((At \cdot \Sigma)^{\leq l} \times X)$ is defined by*

$$\delta_H(\alpha, (u, x)) = \begin{cases} \bot & \alpha \text{ is locked at } u \\ (p, (\mathsf{enq}_l(\alpha p, u), y)) & \delta(\alpha, x) = (p, y) \text{ and } \alpha \text{ is not locked at } u \\ \delta(\alpha, x) & \text{otherwise} \end{cases}$$

where $\alpha \in At$, $p \in \Sigma$, $u \in (At \cdot \Sigma)^{\leq l}$, and $x, y \in X$.

In the third case in the case distinction above, note that if α is not locked at u and $\delta(\alpha, x) \notin \Sigma \times X$, then either $\delta_H(\alpha, x) = \top$ or $\delta_H(\alpha, x) = \bot$.

To prove correctness of the construction of $\mathcal{Q}_H(X, \delta)$, we use two auxiliary lemmas. Lemma 27 characterises the inspection queue during runs, and Lemma 28 relates paths in an automaton (X, δ) to paths in the associated inspection queue automaton $\mathcal{Q}_H(X, \delta)$.

▶ **Lemma 27.** *Let (X, δ) be a GKAT-automaton, let H be a set of word hypotheses, and let $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$. Given $x \in X$ and $n \geq 0$, consider a path*

$$(w_0, x) \xrightarrow{\alpha_0|p_1} (w_1, x_1) \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} (w_n, x_n)$$

in $\mathcal{Q}_H(X, \delta)$. Then $w_n = \mathsf{insp}(l, w_0 \alpha_0 p_1 \cdots \alpha_{n-1} p_n)$.

▶ **Lemma 28.** *Let (X, δ) be a GKAT-automaton, let H be a set of word hypotheses, and let $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$. Suppose there is a path*

$$x_s \xrightarrow{\alpha_0|p_1} \cdots \xrightarrow{\alpha_{n-1}|p_n} x_n \Rightarrow \alpha_n$$

and α_j is not locked at $u\alpha_0 p_1 \cdots \alpha_{j-1}p_j$ for all $0 \leq j \leq n$ and a given $u \in (At \cdot \Sigma)^{\leq l}$. Then the following path exists in $\mathcal{Q}_H(X, \delta)$:

$$(u, x_s) \xrightarrow{\alpha_0|p_1} \cdots \xrightarrow{\alpha_{n-1}|p_n} (\mathsf{insp}(l, u \cdot \alpha_0 p_1 \cdots \alpha_{n-1}p_n), x_n) \Rightarrow \alpha_n.$$

▶ **Theorem 29** (Correctness)**.** *Let H be a set of word hypotheses, (X, δ) be a GKAT automaton, and write $U_H = \{w \mid$ no guarded substring of w violates $H\}$. Then we have*

$$\mathcal{L}(\mathcal{Q}_H(X, \delta), (\varepsilon, x)) = \mathcal{L}((X, \delta), x) \cap U_H \tag{6}$$

Proof. We show (6) by arguing inclusion both ways. Let $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$.

($\subseteq$) In the forward direction, let $w = \alpha_0 p_1 \cdots p_n \alpha_n \in \mathcal{L}(\mathcal{Q}_H(X, \delta), (\varepsilon, x))$. Then by definition, there is a path

$$(\varepsilon, x) \xrightarrow{\alpha_0|p_1} (w_1, x_1) \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} (w_n, x_n) \Rightarrow \alpha_n \tag{7}$$

This implies the existence of a path

$$x \xrightarrow{\alpha_0|p_1} x_1 \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} x_n \Rightarrow \alpha_n$$

in (X, δ), so $w \in \mathcal{L}((X, \delta), x)$. It now suffices to show that there is no guarded substring of w that violates H. Suppose towards a contradiction that there is a guarded substring $\alpha_i p_{i+1} \cdots \alpha_{j-1} p_j \alpha_j$ of w that *does* violate H. Then, since this means there is a word hypothesis in H that $\alpha_i p_{i+1} \cdots \alpha_{j-1} p_j \alpha_j$ violates, then $l \geq j - i$. From Lemma 27 we obtain that $w_j = \mathsf{insp}(l, \alpha_0 p_1 \cdots \alpha_{j-1} p_j)$, and together with $l \geq j - i$ this implies that $\alpha_i p_{i+1} \cdots \alpha_{j-1} p_j$ is a suffix of w_j. Because $\alpha_i p_{i+1} \cdots \alpha_{j-1} p_j \alpha_j$ violates some word hypotheses in H, we get that α_j is locked at w_j. Hence, $\delta_H(\alpha_j, (w_j, x_j)) = \bot$. This contradicts (7), so no such pre-guarded substring can exist. This concludes the forward inclusion.

($\supseteq$) For the reverse inclusion, let $w \in \mathcal{L}((X, \delta), x)$, assume that no guarded substring of w violates H, and write $w = \alpha_0 p_1 \cdots p_n \alpha_n$. Again, immediately, we obtain a path in (X, δ):

$$x \xrightarrow{\alpha_0|p_1} x_1 \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} x_n \Rightarrow \alpha_n.$$

Now, for each $j \leq n$ and $k \in \mathbb{N}$, $\mathsf{insp}(k, \alpha_1 p_1 \cdots \alpha_j p_j)\alpha_{j+1}$ is a subword of w and therefore does not violate H by assumption. By definition, α_{j+1} is not locked at $\alpha_1 p_1 \cdots \alpha_j p_j$. From Lemma 28 it follows that there is the path

$$(\varepsilon, x) \xrightarrow{\alpha_0|p_1} \cdots \xrightarrow{\alpha_{n-1}|p_n} (\mathsf{insp}(l, \alpha_1 p_1 \cdots \alpha_n p_n), x_n) \Rightarrow \alpha_{n+1}$$

in $\mathcal{Q}_H(X, \delta)$. Therefore, w is accepted by the state (ε, x) in $\mathcal{Q}_H(X, \delta)$. ◀

Theorem 29 immediately leads us to the following key result behind our decision procedure.

▶ **Corollary 30.** *Let H be a set of word hypotheses, let $e \in GExp$, and let (X, δ) be a GKAT-automaton with a state $x \in X$ such that $\mathcal{L}(e) = \mathcal{L}((X, \delta), x)$. Then*

$$\mathcal{L}_H(e) = \mathcal{L}(\mathcal{Q}_H(X, \delta), (\varepsilon, x))$$

Proof. Write $U_H = \{w \mid$ no guarded substring of w violates $H\}$. Then, using the definition of $\mathcal{L}_H$, the assumption, and Theorem 29 respectively, in that order,

$$\mathcal{L}_H(e) = \mathcal{L}(e) \cap U_H = \mathcal{L}((X, \delta_X), x) \cap U_H = \mathcal{L}(\mathcal{Q}_H(X, \delta_X), (\varepsilon, x)).$$ ◀

Decidability of word hypotheses. We are now ready to state our decidability result with a complexity bound that improves on the one offered by the decidability-via-translation construction used in the Hoare hypothesis case (see Theorem 19 and Lemma 20). Corollary 30 tells us that to decide whether $\mathcal{L}_H(e) = \mathcal{L}_H(f)$, for $e, f \in \mathit{GExp}$, it suffices to check the language equivalence of states in the inspection queue automata that are constructed from the operational (automaton) semantics of e and f. This is the crux of the improved decision procedure, which proceeds in three steps as follows. Given $e, f \in \mathit{GExp}$,

1. find GKAT-automata (X, δ_X) and (Y, δ_Y) with states $x \in X$ and $y \in Y$ such that $\mathcal{L}(e) = \mathcal{L}((X, \delta_X), x)$ and $\mathcal{L}(f) = \mathcal{L}((Y, \delta_Y), y)$;
2. compute $\mathcal{Q}_H(X, \delta_X)$ and $\mathcal{Q}_H(Y, \delta_Y)$;
3. return the result of deciding $\mathcal{L}(\mathcal{Q}_H(X, \delta_X), (\varepsilon, x)) = \mathcal{L}(\mathcal{Q}_H(Y, \delta_Y), (\varepsilon, y))$.

It is clear from Corollary 30 that this procedure is correct. We are therefore left with analysing the complexity and justifying the claim that it improves on our previous algorithm. We proceed by analysing each of the three steps individually.

Step 1 can be accomplished in a number of different ways [22, 19]. In the cited work, it is shown that for any $e \in \mathit{GExp}$, a GKAT-automaton (X, δ) and a state $x \in X$ can be found in $\mathcal{O}(|e|)$ time such that $\mathcal{L}(e) = \mathcal{L}((X, \delta), x)$ and $|X| \in \mathcal{O}(|e|)$. Therefore, if we define $n = |e| + |f|$, then Step 1 can be completed in $\mathcal{O}(n)$ time and the automata (X, δ_X) and (Y, δ_Y) that were found have $\mathcal{O}(n)$ many states each.

For Step 2, we obviously need to assume that H is finite (this assumption will appear in the statement of Theorem 31 below). Let $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$, $m = |At|$, $k = |\Sigma|$, and write N for the total number of states in $\mathcal{Q}_H(X, \delta_X)$ and $\mathcal{Q}_H(Y, \delta_Y)$. We compute N explicitly and provide it an asymptotic bound as follows:

$$\begin{aligned} N &= |(At \cdot \Sigma)^{\leq l} \times X| + |(At \cdot \Sigma)^{\leq l} \times Y| \\ &= |(At \cdot \Sigma)^{\leq l}|(|X| + |Y|) \\ &= \Big(\frac{(mk)^{l+1} - 1}{mk - 1}\Big)(|X| + |Y|) \\ &\in \mathcal{O}((mk)^l n) \end{aligned} \tag{8}$$

In the last step, we let $n = |e| + |f|$ and from the discussion above we know $|X| + |Y| \in \mathcal{O}(n)$.

For Step 3, it is observed that inspection queue automata are equivalent to ordinary GKAT automata, and therefore we can use the original decision algorithm for GKAT presented in [22]. In [22] an algorithm of Hopcroft and Karp [7] is adapted to decide language equivalence of states of GKAT-automata in nearly-linear time (when the number of primitive tests is held constant) with respect to the total number of states in the GKAT-automata involved. Here, *nearly-linear* means $\mathcal{O}(n\alpha(n))$ time, with n the size parameter, in which α denotes the inverse of the Ackermann function. For us, the number of states involved is N, calculated above in (8), so Step 3 is completed in $\mathcal{O}(N\alpha(N))$ many steps. Since $N \in \mathcal{O}((mk)^l n)$, this is bounded by $\mathcal{O}((mk)^l n \cdot \alpha((mk)^l n))$.[1]

Putting all of this together, we obtain the following decidability result.

▶ **Theorem 31** (Decidability of word hypotheses). *Let $e, f \in \mathit{GExp}$, let H be a finite set of word hypotheses, and let $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$ (note that l is necessarily finite). Let $n = |e| + |f|$, $m = |At|$, and $k = |\Sigma|$. Then it is decidable in $\mathcal{O}((mk)^l n \cdot \alpha((mk)^l n))$ time whether $\mathcal{L}_H(e) = \mathcal{L}_H(f)$.*

[1] There is a subtle point here that is easy to miss. It is not trivial to conclude from $f(n) \in \mathcal{O}(g(n))$ that $\alpha(f(n)) \in \mathcal{O}(\alpha(g(n)))$, although this implication is true.

Recall that the complexity bound in Theorem 3 holds the number of primitive tests constant. We are also assuming that the primitive tests are held constant in Theorem 31 (and in Corollary 32 below), as well as the number of primitive programs, but we allow l to vary. The presence of m and k in the complexity bound in Theorem 31 are necessary as they form the base of the exponential factor in the runtime.

Returning to the content of earlier sections, observe that a Hoare hypothesis is a word hypothesis of length 1. In the special case where H is a set of Hoare hypotheses, therefore, $l = 1$ in Theorem 31. This immediately leads us to the following corollary of Theorem 31, which improves on the complexity bound given in Lemma 20.

▶ **Corollary 32.** *Let H be a set of Hoare hypotheses and let $e, f \in GExp$. Then it is decidable in $\mathcal{O}(n\alpha(n))$ time where $n = |e| + |f|$, whether $\mathcal{L}_H(e) = \mathcal{L}_H(f)$.*

The presented decision procedure shows that word hypotheses can be added to GKAT whilst maintaining the nearly linear time complexity of deciding equivalence.

▶ Remark 33. The decision procedure relies on an automaton construction (Definition 26) which remembers in the queue *all* of the last l actions and atoms, where l is the maximum length of hypotheses in H. If H contains only a few (possibly long) hypotheses, then this may not be optimal, and instead one could choose only to store pre-guarded strings that are relevant for some hypothesis in H. This may reduce the size of the automaton, but it also complicates the definition (and computation) of the transition function. In particular, one then needs to remember all pre-guarded strings that are a prefix of some hypothesis in H, and in fact it can be that only a *suffix* of the current queue state is a prefix of a hypothesis in H. We instead chose to present the conceptually simpler construction in (Definition 26).

Another alternative construction may be to reduce word hypotheses to Hoare hypotheses, through the introduction of fresh variables. These variables can then uniquely identify a prefix of a hypothesis, and in this way implicitly store these in a queue which then only needs to be of length 1 (at the cost of introducing fresh variables). Similarly to the above-mentioned construction, this may work better when there are few hypotheses in H, but also complicates the analysis. We believe both alternatives are interesting for consideration in future work, especially in the context of an implementation.

5 Future work

We have provided an extension of GKAT with Hoare hypotheses and word hypotheses. The introduced semantics for Hoare hypotheses is proved to be complete and decidable in Section 3. In Section 4, we generalised Hoare hypotheses to word hypotheses, and provided a decision procedure that also covers the plain Hoare hypotheses, and is more efficient in that case. The decision procedure makes use of an automata-theoretic construction.

An open question is completeness of GKAT with word hypotheses. A possible proof strategy would be to reduce word hypotheses to Hoare hypotheses via fresh variables, as discussed in Remark 33, and then reuse the completeness result of Hoare hypotheses. This would require, among other things, a proof that a derivation using Hoare hypotheses with fresh variables can be translated to a derivation with the equivalent word hypotheses. This question is beyond the scope of the current paper but seems fruitful for future research.

A natural question is whether more general hypotheses can be added to GKAT, like in KA. This seems very challenging as there is no canonical interpretation of hypotheses like what exists for language models of KA. An important difference is that for language models of KA the addition of hypotheses expands the original language semantics, whereas

our GKAT semantics with hypotheses shrinks the plain GKAT semantics. A semantics for GKAT with hypotheses that expands the original semantics seems unlikely, as it is unclear how to remain deterministic. On the other hand, to develop KA with hypotheses based on a semantics that shrinks languages seems a relevant future direction.

A more straightforward question is how Hoare and word hypotheses can be added to bisimulation GKAT [19], which is a process-theoretic variation of GKAT that lacks the *late-termination axiom* $e0 = 0$. A related question is whether we can obtain language GKAT as an instance of bisimulation GKAT with the added hypothesis $e0 = 0$.

References

1 Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. NetKAT: semantic foundations for networks. In *POPL*, pages 113–126. ACM, 2014. doi:10.1145/2535838.2535862.

2 E. Cohen. Hypotheses in Kleene algebra. Technical report, Bellcore, Morristown, NJ, 1994.

3 Ernie Cohen, Dexter Kozen, and Frederick Smith. The complexity of Kleene algebra with tests. Technical report, Cornell University, 1996.

4 Amina Doumane, Denis Kuperberg, Damien Pous, and Cécilia Pradic. Kleene algebra with hypotheses. In *FoSSaCS*, volume 11425 of *Lecture Notes in Computer Science*, pages 207–223. Springer, 2019. doi:10.1007/978-3-030-17127-8_12.

5 Leandro Gomes, Patrick Baillot, and Marco Gaboardi. A Kleene algebra with tests for union bound reasoning about probabilistic programs. In *CSL*, volume 326 of *LIPIcs*, pages 35:1–35:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.CSL.2025.35.

6 C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, October 1969. doi:10.1145/363235.363259.

7 John E. Hopcroft and Richard M. Karp. A linear algorithm for testing equivalence of finite automata. Technical Report TR 71-114, Cornell University, 1971. URL: https://hdl.handle.net/1813/5958.

8 Spencer Van Koevering, Wojciech Rozowski, and Alexandra Silva. Weighted GKAT: completeness and complexity. In *ICALP*, volume 334 of *LIPIcs*, pages 172:1–172:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.172.

9 Dexter Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997. doi:10.1145/256167.256195.

10 Dexter Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. Comput. Log.*, 1(1):60–76, 2000. doi:10.1145/343369.343378.

11 Dexter Kozen and Konstantinos Mamouras. Kleene algebra with equations. In *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 280–292. Springer, 2014. doi:10.1007/978-3-662-43951-7_24.

12 Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In *CSL*, volume 1258 of *Lecture Notes in Computer Science*, pages 244–259. Springer, 1996. doi:10.1007/3-540-63172-0_43.

13 Jack Liell-Cock and Sam Staton. Compositional imprecise probability: A solution from graded monads and markov categories. *Proc. ACM Program. Lang.*, 9(POPL):1596–1626, 2025. doi:10.1145/3704890.

14 Yoshiki Nakamura. Undecidability of the positive calculus of relations with transitive closure and difference: Hypothesis elimination using graph loops. In *RAMiCS*, volume 14787 of *Lecture Notes in Computer Science*, pages 207–224. Springer, 2024. doi:10.1007/978-3-031-68279-7_13.

15 Damien Pous. Symbolic algorithms for language equivalence and Kleene algebra with tests. In *POPL*, pages 357–368. ACM, 2015. doi:10.1145/2676726.2677007.

16 Damien Pous, Jurriaan Rot, and Jana Wagemaker. On tools for completeness of Kleene algebra with hypotheses. *Log. Methods Comput. Sci.*, 20(2), 2024. doi:10.46298/LMCS-20(2:8)2024.

17 Wojciech Rozowski, Tobias Kappé, Dexter Kozen, Todd Schmid, and Alexandra Silva. Probabilistic guarded KAT modulo bisimilarity: Completeness and complexity. In *ICALP*, volume 261 of *LIPIcs*, pages 136:1–136:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.136.

18 Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966. doi:10.1145/321312.321326.

19 Todd Schmid, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: Coequations, coinduction, and completeness. In *ICALP*, volume 198 of *LIPIcs*, pages 142:1–142:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.142.

20 Todd Schmid, Tobias Kappé, and Alexandra Silva. A complete inference system for skip-free guarded Kleene algebra with tests. In *ESOP*, volume 13990 of *Lecture Notes in Computer Science*, pages 309–336. Springer, 2023. doi:10.1007/978-3-031-30044-8_12.

21 Todd Schmid, Wojciech Rozowski, Alexandra Silva, and Jurriaan Rot. Processes parametrised by an algebraic theory. In *ICALP*, volume 229 of *LIPIcs*, pages 132:1–132:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.132.

22 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time. *Proc. ACM Program. Lang.*, 4(POPL):61:1–61:28, 2020. doi:10.1145/3371129.

23 Eugene W. Stark and Scott A. Smolka. A complete axiom system for finite-state probabilistic processes. In Gordon D. Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 571–596. The MIT Press, 2000.

24 Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, April 1975. doi:10.1145/321879.321884.

25 Jacob Wasserstein. Guarded NetKAT: Soundness, partial-completeness, and decidability, 2023. Master thesis, Cornell University.

26 Cheng Zhang, Qiancheng Fu, Hang Ji, Ines Santacruz Del Valle, Alexandra Silva, and Marco Gaboardi. Outrunning big KATs: Efficient decision procedures for variants of GKAT. In *ESOP (2)*, volume 16502 of *Lecture Notes in Computer Science*, pages 392–423. Springer, 2026. doi:10.1007/978-3-032-22723-2_14.

27 Cheng Zhang, Tobias Kappé, David E. Narváez, and Nico Naus. CF-GKAT: efficient validation of control-flow transformations. *Proc. ACM Program. Lang.*, 9(POPL):600–626, 2025. doi:10.1145/3704857.

A Additional proofs for Section 3

In the proofs below we identify bc and $b \wedge c$, implicitly using axiom (S6) from Figure 1. We begin by observing the following consequence of the GKAT axioms (see Figure 1):

$$\mathsf{GKAT} \vdash b(e +_b f) = be \tag{9}$$

We call this *restriction*, and it can be derived as follows. Let us start with an identity we will call (U4'), which can be shown using (U2), (U4), and (U2), in that order:

$$e +_b f = f +_{\bar{b}} e = \bar{b}f +_{\bar{b}} e = e +_b \bar{b}f \tag{U4'}$$

We also observe that each test $b \in \mathit{BExp}$ can really be thought of as syntactic sugar for $1 +_b 0$, what we will call (B). The equation (B) is due to (U1), (BA), (U4), (U4'), and (BA) in that order:

$$b = b +_b b = b1 +_b b = 1 +_b b = 1 +_b \bar{b}b = 1 +_b 0 \tag{B}$$

Finally, the identity (restriction) is derived as follows:

$$\begin{aligned}
b(e +_b f) &= (1 +_b 0)(e +_b f) && \text{(B)}\\
&= (e +_b f) +_b 0 && \text{(U5, S4, S2)}\\
&= e +_b (f +_b 0) && \text{(U3, BA)}\\
&= e +_b (1 +_b 0)f && \text{(S2, S4, U5)}\\
&= e +_b bf && \text{(B)}\\
&= e +_b \bar{b}bf && \text{(U4')}\\
&= e +_b 0 && \text{(BA, S2)}\\
&= (1 +_b 0)e && \text{(S2, S4, U5)}\\
&= be && \text{(B)}
\end{aligned}$$

▶ **Lemma 7.** *Let $e, f \in GExp$.* $\mathsf{GKAT} + \mathsf{Hst}_{p,b} \vdash e = f$ *if and only if* $\mathsf{GKAT} + \mathsf{Inv}_{p,b} \vdash e = f$.

Proof. For the right-to-left direction, we have the following derivation of $p(e +_b f) = pe +_b pf$:

$$\begin{aligned}
\mathsf{GKAT} + \mathsf{Inv}_{p,b} \vdash \quad p(e +_b f) &= p(e +_b f) +_b p(e +_b f) && \text{(U1)}\\
&= bp(e +_b f) +_b \bar{b}p(e +_b f) && \text{(U4, U4')}\\
&= bpb(e +_b f) +_b \bar{b}p\bar{b}(e +_b f) && (\mathsf{Inv}_{p,b})\\
&= bpbe +_b \bar{b}p\bar{b}f && \text{(restriction, U2)}\\
&= bpe +_b \bar{b}pf && (\mathsf{Inv}_{p,b})\\
&= pe +_b pf && \text{(U4, U4')}
\end{aligned}$$

In other words, $\mathsf{Inv}_{p,b}$ implies that p can be hoisted from any conditional $e +_b f$.

Conversely, the hoisting hypothesis $\mathsf{Hst}_{p,b}$ implies the invariant test hypothesis $\mathsf{Inv}_{p,b}$. To see this, choose $e = 1$ and $f = 0$ and observe that

$$\begin{aligned}
\mathsf{GKAT} + \mathsf{Hst}_{p,b} \vdash \quad bpb &= bp(1 +_b 0) && \text{(B)}\\
&= b(p1 +_b p0) && (\mathsf{Hst}_{p,b})\\
&= bp1 && \text{(restriction)}\\
&= bp
\end{aligned}$$

The same argument with $e = 0$ and $f = 1$ shows that the hoisting principle for p over b implies $\bar{b}p = \bar{b}p\bar{b}$. ◀

▶ **Lemma 8.** *Let $e, f \in GExp$.* $\mathsf{GKAT} + \mathsf{Inv}_{p,b} \vdash e = f$ *if and only if* $\mathsf{GKAT} + \mathsf{Com}_{p,b} \vdash e = f$.

Proof. In the forward direction,

$$\begin{aligned}
\mathsf{GKAT} + \mathsf{Inv}_{p,b} \vdash pb &= pb +_b pb && \text{(U1)}\\
&= bpb +_b \bar{b}pb && \text{(U4, U4')}\\
&= bp +_b \bar{b}p\bar{b}b && (\mathsf{Inv}_{p,b})\\
&= bp +_b \bar{b}p0 && \text{(BA)}\\
&= bp +_b 0 && \text{(S3)}\\
&= bp && \text{(U4, S2, S4, U5, B)}
\end{aligned}$$

The reverse direction is much simpler:[2]

$$\begin{aligned} \mathsf{GKAT} + \mathsf{Com}_{p,b} \vdash bpb &= bbp && (bp = pb) \\ &= bp && (\mathsf{BA}) \end{aligned}$$

Similar derivations can be given for $\bar{b}p = \bar{b}p\bar{b}$ and $\bar{b}p = p\bar{b}$. ◀

▶ **Lemma 10.** *For any set of Hoare hypotheses H, $e, f \in GExp$ and $b \in BExp$:*

$$\mathcal{L}_H(b) = \{\alpha \in At \mid \alpha \leq_{\mathsf{BA}} b\} \qquad \mathcal{L}_H(ef) = \mathcal{L}_H(e) \diamond \mathcal{L}_H(f)$$
$$\mathcal{L}_H(e +_b f) = b \diamond \mathcal{L}_H(e) \cup \bar{b} \diamond \mathcal{L}_H(f) \qquad \mathcal{L}_H(e^{(b)}) = (b \diamond \mathcal{L}_H(e))^\star \diamond \bar{b}$$

Proof. Note that $\mathcal{L}_H(b) = \mathcal{L}(b)$ follows easily from the definition of $\mathcal{L}_H$, giving the first case. For the remaining cases, we first define

$$X = \{\alpha_1 p_1 \cdots \alpha_n p_n \alpha_{n+1} \mid (bp_i = bp_i c \in H \wedge \alpha_i \leq b) \Rightarrow \alpha_{i+1} \leq c\}$$

and observe that for any $e \in GExp$, $\mathcal{L}_H(e) = \mathcal{L}(e) \cap X$.

We will need the following identity: for any guarded languages L_1, L_2, we have

$$(L_1 \diamond L_2) \cap X = (L_1 \cap X) \diamond (L_2 \cap X) \tag{$\dagger$}$$

The identity ($\dagger$) above is derived from the fact that we have a guarded string $u\alpha v \in X$ if and only if we have guarded strings $u\alpha \in X$ and $\alpha v \in X$. In particular, one should note that $At \subseteq X$, so from ($\dagger$) we also obtain $(b \diamond L) \cap X = b \diamond (L \cap X)$ and $(L \diamond b) \cap X = (L \cap X) \diamond b$ for any guarded language L and $b \in BExp$.

In the guarded union case, we have

$$\begin{aligned} \mathcal{L}_H(e +_b f) &= (b \diamond \mathcal{L}(e) \cup \bar{b} \diamond \mathcal{L}(f)) \cap X \\ &= ((b \diamond \mathcal{L}(e)) \cap X) \cup ((\bar{b} \diamond \mathcal{L}(f)) \cap X) \\ &= (b \diamond (\mathcal{L}(e) \cap X)) \cup (\bar{b} \diamond (\mathcal{L}(f) \cap X)) && (\dagger) \\ &= (b \diamond \mathcal{L}_H(e) \cup (\bar{b} \diamond \mathcal{L}_H(f)) \end{aligned}$$

For concatenation, we have

$$\begin{aligned} \mathcal{L}_H(e \cdot f) &= (\mathcal{L}(e) \diamond \mathcal{L}(f)) \cap X \\ &= (\mathcal{L}(e) \cap X) \diamond (\mathcal{L}(f) \cap X) && (\dagger) \\ &= \mathcal{L}_H(e) \diamond \mathcal{L}_H(f) && (\text{def. of } \mathcal{L}_H) \end{aligned}$$

For the star case, we first derive the following for all $n \geq 0$:

$$(b \diamond L)^{\langle n \rangle} \cap X = (b \diamond (L \cap X))^{\langle n \rangle} \tag{10}$$

We proceed by induction on n. The base case is trivial as $At \subseteq X$. For $n+1$ we derive

$$\begin{aligned} (b \diamond L)^{\langle n+1 \rangle} \cap X &= ((b \diamond L) \diamond (b \diamond L)^{\langle n \rangle}) \cap X \\ &= ((b \diamond L) \cap X) \diamond ((b \diamond L)^{\langle n \rangle} \cap X) && (\dagger) \end{aligned}$$

[2] Interestingly, it appears that the use of (S3) is necessary in the forward direction, but the reverse direction does not require (S3).

$$
\begin{aligned}
&= (b \diamond (L \cap X)) \diamond ((b \diamond L)^{\langle n \rangle} \cap X) && (\dagger)\\
&= (b \diamond (L \cap X)) \diamond (b \diamond (L \cap X))^{\langle n \rangle} && (\text{I.H.})\\
&= (b \diamond (L \cap X))^{\langle n+1 \rangle}
\end{aligned}
$$

Returning to the $(-)^{(b)}$ case, recall that for a guarded language L, $L^\star = \bigcup_{n \geq 0} L^{\langle n \rangle}$. We have

$$
\begin{aligned}
\mathcal{L}_H(e^{(b)}) &= ((b \diamond \mathcal{L}(e))^\star \diamond \bar{b}) \cap X\\
&= ((b \diamond \mathcal{L}(e))^\star \cap X) \diamond \bar{b} && (\dagger)\\
&= \left(\left(\bigcup_{n \geq 0} (b \diamond \mathcal{L}(e))^{\langle n \rangle} \right) \cap X \right) \diamond \bar{b}\\
&= \left(\bigcup_{n \geq 0} \left((b \diamond \mathcal{L}(e))^{\langle n \rangle} \cap X \right) \right) \diamond \bar{b}\\
&= \left(\bigcup_{n \geq 0} (b \diamond (\mathcal{L}(e) \cap X))^{\langle n \rangle} \right) \diamond \bar{b} && (10)\\
&= (b \diamond (\mathcal{L}(e) \cap X))^\star \diamond \bar{b}\\
&= (b \diamond \mathcal{L}_H(e))^\star \diamond \bar{b}
\end{aligned}
$$

◀

▶ **Theorem 11** (Soundness). *Let H be a set of Hoare hypotheses. For any $e, f \in GExp$, if $\mathsf{GKAT} + H \vdash e = f$, then $\mathcal{L}_H(e) = \mathcal{L}_H(f)$.*

Proof. The derivation relation is the smallest congruence relation that contains the axioms of GKAT and the equalities in H. By showing that the language equivalence relation is a congruence that contains the axioms of GKAT and the equalities in H, it follows that $\mathsf{GKAT} + H \vdash e = f$ implies $\mathcal{L}_H(e) = \mathcal{L}_H(f)$. If $e = f$ is an axiom of GKAT, we obtain from GKAT soundness that $\mathcal{L}(e) = \mathcal{L}(f)$. For $e = f \in H$, we know that $e = bp$ and $f = bpc$, and the result follows immediately.

It remains to show that language equivalence up to H is a congruence. Reflexivity, symmetry and transitivity are trivial. It remains to show closure under each of the operators. To this end, let $\mathcal{L}_H(e_0) = \mathcal{L}_H(f_0)$ and $\mathcal{L}_H(e_1) = \mathcal{L}_H(f_1)$. Then

$$\mathcal{L}_H(e_0 +_b f_0) = b \diamond \mathcal{L}_H(e_0) \cup \bar{b} \diamond \mathcal{L}_H(f_0) = b \diamond \mathcal{L}_H(e_1) \cup \bar{b} \diamond \mathcal{L}_H(f_1) = \mathcal{L}_H(e_1 +_b f_1)$$

using Lemma 10. The cases $\mathcal{L}_H(e_0 \diamond f_0) = \mathcal{L}_H(e_1 \diamond f_1)$ and $\mathcal{L}_H(e_0^{(b)}) = \mathcal{L}_H(f_0^{(b)})$ follow similarly from Lemma 10. ◀

▶ **Lemma 20.** *Let $e \in GExp$ and let H be a set of Hoare hypotheses. Then*

$$|T_H(e)| = \sum_{p \in e} (3(2^{n_p^H}) - 3) + |e|$$

Proof. We proceed by induction on e. If $e = b \in BExp$, the result trivially holds because $T_H(b) = b$.

For $e = p \in \Sigma$, we proceed to prove that $|T_H(p)| = 3(2^{n_p^H}) - 2$ by induction on the number of hypotheses in H that involve p. If there exists no hypothesis in H that involves p, the result trivially holds. Otherwise pick any $h \in H$ of the form $bp = bpc$ for some $b, c \in BExp$. As the order of reductions in T_H does not matter (Lemma 15, and it is also easy to see it does not affect the size), we assume the reduction T_h is performed "last", i.e. $T_H(e) = T_h(T_{H \setminus h}(e))$. To alleviate notation, we denote n_p^H with n and $n_p^{H \setminus h}$ with m. Observe that $n = m + 1$. Our induction hypothesis states that $|T_{H \setminus h}(p)| = 3(2^m) - 2$. We derive

$$
\begin{aligned}
|T_H(p)| &= |T_h(T_{H\setminus h}(p))| \\
&= |T_{H\setminus h}(T_h(p))| \\
&= |T_{H\setminus h}(pc +_b p)| \\
&= |T_{H\setminus h}(p)c +_b T_{H\setminus h}(p)| \\
&= 2|T_{H\setminus h}(p)| + 2 \\
&= 2(3(2^m) - 2) + 2 && \text{(Induction Hypothesis)} \\
&= 3(2^{m+1}) - 2 = 3(2^n) - 2 && (2(2^m) = 2^{m+1},\ n = m+1)
\end{aligned}
$$

For $e +_b f$ we abbreviate n_p^H with n_p and derive

$$
\begin{aligned}
&|T_H(e +_b f)| \\
&= |T_H(e) +_b T_H(f)| \\
&= |T_H(e)| + 1 + |T_H(f)| \\
&= \sum_{p\in e}(3(2^{n_p}) - 3) + |e| + 1 + \sum_{p\in f}(3(2^{n_p}) - 3) + |f| && \text{(Induction Hypothesis)} \\
&= \sum_{p\in e+_b f}(3(2^{n_p^H}) - 3) + |e +_b f|
\end{aligned}
$$

The other inductive cases follow similarly. ◀

B Additional proofs for Section 4

▶ **Lemma 27.** *Let (X, δ) be a GKAT-automaton, let H be a set of word hypotheses, and let $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$. Given $x \in X$ and $n \geq 0$, consider a path*

$$(w_0, x) \xrightarrow{\alpha_0|p_1} (w_1, x_1) \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} (w_n, x_n)$$

in $\mathcal{Q}_H(X, \delta)$. Then $w_n = \mathsf{insp}(l, w_0\alpha_0p_1 \cdots \alpha_{n-1}p_n)$.

Proof. We proceed by induction on n. For $n = 0$, $\mathsf{insp}(l, w_0) = w_0$ follows because $w_0 \in (At \cdot \Sigma)^{\leq l}$.

For $n + 1$, we derive

$$
\begin{aligned}
w_{n+1} &= \mathsf{enq}_l(\alpha_{n+1}p_{n+1}, w_n) && \text{(def. of } \delta_H) \\
&= \mathsf{insp}(l, w_n\alpha_{n+1}p_{n+1}) && \text{(def. of } \mathsf{enq}_l) \\
&= \mathsf{insp}(l, w_0\alpha_1p_1 \cdots \alpha_{n+1}p_{n+1}) && \text{(ind. hyp.)}
\end{aligned}
$$

as desired. This concludes the proof. ◀

▶ **Lemma 28.** *Let (X, δ) be a GKAT-automaton, let H be a set of word hypotheses, and let $l = \max\{\mathsf{len}(w) \mid (w = wc) \in H\}$. Suppose there is a path*

$$x_s \xrightarrow{\alpha_0|p_1} \cdots \xrightarrow{\alpha_{n-1}|p_n} x_n \Rightarrow \alpha_n$$

and α_j is not locked at $u\alpha_0p_1 \cdots \alpha_{j-1}p_j$ for all $0 \leq j \leq n$ and a given $u \in (At \cdot \Sigma)^{\leq l}$. Then the following path exists in $\mathcal{Q}_H(X, \delta)$:

$$(u, x_s) \xrightarrow{\alpha_0|p_1} \cdots \xrightarrow{\alpha_{n-1}|p_n} (\mathsf{insp}(l, u \cdot \alpha_0p_1 \cdots \alpha_{n-1}p_n), x_n) \Rightarrow \alpha_n.$$

Proof. We proceed by induction on n. If $n = 0$, then α_0 is not locked at u by assumption. Hence, $\delta_H(\alpha_0, (u, x_s)) = \delta(\alpha_0, x) = \top$, and we can conclude.

For the induction step, let $n > 0$ and consider the path

$$x_1 \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} x_n \Rightarrow \alpha_n$$

starting at α_1. We relabel by setting $\beta_i = \alpha_{i+1}$ for $i \in \{0, \dots n-1\}$ and $q_i = p_{i+1}$ for $i \in \{1, \dots n-1\}$. This gives the path

$$x_1 \xrightarrow{\beta_0|q_1} \cdots \xrightarrow{\beta_{n-2}|q_{n-1}} x_n \Rightarrow \beta_{n-1}$$

Now, for all $n \in \mathbb{N}$, the guarded string

$$\mathsf{insp}(n, u\alpha_0 p_1 \alpha_1 p_2 \cdots \alpha_{j-1} p_j) \cdot \alpha_j$$

does not violate H for all $0 \leq j \leq n$ by assumption. Consequently, for all $n \in \mathbb{N}$, the string

$$\mathsf{insp}(n, u\alpha_0 p_1 \beta_0 q_1 \cdots \beta_{k-2} q_{k-1}) \cdot \beta_{k-1}$$

does not violate H for any $1 \leq k \leq n$. Hence, β_{k-1} is not locked at $\mathsf{insp}(l, u\alpha_0 p_1 \beta_0 q_1 \cdots \beta_{k-1} q_{k-1})$ for any $1 \leq k \leq n-1$. Then we can apply the induction hypothesis to obtain a path,

$$(u\alpha_0 p_1, x_1) \xrightarrow{\beta_0|q_1} \cdots \xrightarrow{\beta_{n-2}|q_{n-1}} (\mathsf{insp}(l, u \cdot \alpha_0 p_1 \beta_0 q_1 \cdots \beta_{n-2} q_{n-1}), x_n) \Rightarrow \beta_{n-1}$$

If we reverse our labelling, we obtain the path

$$(u\alpha_0 p_1, x_1) \xrightarrow{\alpha_1|p_2} \cdots \xrightarrow{\alpha_{n-1}|p_n} (\mathsf{insp}(l, u \cdot \alpha_0 p_1 \alpha_1 p_2 \cdots \alpha_{n-1} p_n), x_n) \Rightarrow \alpha_n$$

As $\delta(\alpha_0, x_s) = (p_1, x_1)$ and α_0 is not locked at u by assumption, we get that $\delta_H(\alpha_0, u, x_s) = (p_1, \mathsf{insp}(l, u\alpha_0 p_1), x_1)$, and we can conclude. ◀