Abstract. The logic of knowledge and justified assumptions, also known as logic of grounded knowledge (GK), was proposed by Lin and Shoham as a general logic for nonmonotonic reasoning. To date, it has been used to embed it default logic (propositional case), autoepistemic logic, Turner’s logic of universal causation, and general logic programming under stable model semantics. Besides showing the generality of GK as a logic for nonmonotonic reasoning, these embeddings shed light on the relationships among these other logics.

In this paper, for the first time, we show how the logic of GK can be embedded into disjunctive logic programming in a polynomial but non-modular translation with new variables. The result can then be used to compute the extension/expansion semantics of default logic, autoepistemic logic and Turner’s logic of universal causation by disjunctive ASP solvers such as claspD(-2), DLV, GNT and cmodels.

1 Introduction

Lin and Shoham [21] proposed a logic with two modal operators K and A, standing for knowledge and assumption, respectively. The idea is that one starts with a set of assumptions (those true under the modal operator A), computes the minimal knowledge under this set of assumptions, and then checks to see if the assumptions were justified in that they agree with the resulting minimal knowledge. For instance, consider the GK formula A p ⊨ K p. If we assume p, then we can conclude that we know p, thus the assumption that p holds is justified, and we get a GK model where both A p and K p are true. (There is another GK model where we do not assume p and hence do not know p.) However, there is no GK model of ¬A p ⊨ K p: if we do not assume p, we are forced to conclude K p, but then knowledge and assumptions do not coincide; if we do assume p, we cannot conclude that we know p and thus assuming p was not justified.

To date, there have been embeddings from default logic [30] and autoepistemic logic [27] to the logic of GK [21], from Turner’s logic of universal causation [36] to the logic of GK [14], as well as from formal programs [9] to the logic of GK [22]. Among other things, these embeddings shed new light on nonmonotonic reasoning, and have led to an interesting characterization of strong equivalence and thus assuming p was not justified.

In this paper, for the first time, we consider computing models of GK theories by disjunctive logic programs. We shall propose a polynomial translation from a (pure) GK theory to a disjunctive logic program such that there is a one-to-one correspondence between GK models of the GK theory and answer sets of the resulting disjunctive logic program. The result can then be used to compute the extension/expansion semantics of default logic, autoepistemic logic and Turner’s logic of universal causation by disjunctive ASP solvers such as claspD [7], claspD-2 [10], DLV [18], GNT [12] and cmodels [11]. To substantiate this claim, we have implemented the translation and report on some experiments that we conducted on the special case of computing extensions for Reiter’s default logic [30]. The implementation, called gk2dlp, is available for download from the second author’s home page at http://informatik.uni-leipzig.de/~strass/gk2dlp/.

Providing implementations for theoretical formalisms has a long tradition in nonmonotonic reasoning, for an overview see [6]. In fact, nonmonotonic reasoning itself originated from a desire to more accurately model the way humans reason, and was since its conception driven by applications in commonsense reasoning [25, 26]. Today, thanks to extensive research efforts, we know how closely interrelated the different formalisms for nonmonotonic reasoning are, and can use this knowledge to improve the scope of implementations.

This paper is organized as follows. Section 2 reviews logic programs, the logic of GK and default and autoepistemic logics. Section 3 presents our main result, the mapping from GK to disjunctive logic programming; due to space constraints, we could however not include any of the proofs. Section 4 presents our prototypical implementation, several experiments we conducted to analyze the translation, possible applications for it as well as a comparison with previous related work. Section 5 concludes with ideas for future work.

2 Preliminaries

We assume a propositional language with two zero-place logical connectives ⊤ for tautology and ⊥ for contradiction. We denote by Atom the set of atoms, the signature of our language, and Lit the set of literals: Lit = Atom ∪ {¬p | p ∈ Atom}. A set I of literals is called complete if for each atom p, exactly one of {p, ¬p} is in I.

In this paper, we identify an interpretation with a complete set of literals. If I is a complete set of literals, we use it as an interpretation when we say that it is a model of a formula, and we use it as a set of literals when we say that it entails a formula. In particular, we denote by Th(I) the logical closure of I (considered to be a set of literals).

2.1 Logic Programming

A nested expression is built from literals using the 0-place connectives ⊤ and ⊥, the unary connective “not” and the binary connectives “¬” and “·” for conjunction and disjunction. A logic program with nested expressions is a finite set of rules of the form F ← G, where F and G are nested expressions. The answer set of a logic

1 School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
2 Computer Science Institute, Leipzig University, Leipzig, Germany
F expression program with nested expressions is defined as in 19. Given a nested expression F and a set S of literals, we define when S satisfies F, written \( S \models F \) below, recursively as follows (l is a literal):

\[
\begin{align*}
S & \models l \text{ if } l \in S, \\
S & \models \top \text{ and } S \not\models \bot, \\
S & \not\models F \text{ if } S \not\models F, \\
S & \models F \land G \text{ if } S \models F \text{ and } S \models G, \text{ and} \\
S & \models F \lor G \text{ if } S \models F \text{ or } S \models G.
\end{align*}
\]

S satisfies a rule \( F \rightarrow G \) if \( S \models F \) whenever \( S \models G \). S satisfies a logic program \( P \), written \( S \models P \), if \( S \) satisfies all rules in \( P \).

\[
\text{The reduct} P^0 \text{ of } P \text{ related to } S \text{ is the result of replacing every maximal subexpression of } P \text{ that has the form not } F \text{ with } \bot \text{ if } S \models F, \text{ and with } \top \text{ otherwise. For a logic program } P \text{ without not, the answer set of } P \text{ is any minimal consistent subset } S \text{ of } \text{Lit} \text{ that satisfies } P. \text{ We use } \Gamma_P(S) \text{ to denote the set of answer sets of } P^0. \text{ Now a consistent set } S \text{ of literals is an answer set of } P \text{ iff } S \in \Gamma_P(S). \text{ Every logic program with nested expressions can be equivalently translated to disjunctive logic programs with disjunctive rules of the form}
\]
\[
\begin{align*}
& l_1 \land \cdots \land l_k \leftarrow l_{k+1}, \ldots, l_t, \text{ not } l_{t+1}, \ldots, \text{ not } l_m, \\
& \text{ not not } l_{m+1}, \ldots, \text{ not not } l_n
\end{align*}
\]

where \( n \geq m \geq t \geq k \geq 0 \) and \( l_1, \ldots, l_n \) are propositional literals.

2.2 Default Logic

Default logic [30] is for closing gaps in incomplete knowledge bases. This is done by defaults, that allow to express rules of thumb such as “birds usually fly” and “tools usually work.” For a given logical language, a default is any expression of the form \( \phi \leftarrow \psi_1, \ldots, \psi_n / \varphi \) where \( \phi, \psi_1, \ldots, \psi_n, \varphi \) are formulas of the underlying language. A default theory is a pair \( (W, D) \), where \( W \) is a set of formulas and \( D \) is a set of defaults. The meaning of default theories is given through the notion of extensions. An extension of a default theory \( (W, D) \) is “interpreted as an acceptable set of beliefs that one may hold about the incompletely specified world \( W \)” [30]. For a default theory \( (W, D) \) and any set \( S \) of formulas let \( \Gamma(S) \) be the smallest set satisfying (1) \( W \subseteq \Gamma(S) \), (2) \( \text{Th}(\Gamma(S)) = \Gamma(S) \), (3) \( \phi \leftrightarrow \psi_1, \ldots, \psi_n / \varphi \in D \), \( \phi \in \Gamma(S) \) and \( \neg \psi_1, \ldots, \neg \psi_n \notin S \), then \( \varphi \in \Gamma(S) \). A set \( E \) of formulas is called an extension for \( (W, D) \) if \( \Gamma(E) = E \).

2.3 Autoepistemic Logic

Moore [27] strives to formalize an ideally rational agent reasoning about its own beliefs. He uses a belief modality \( L \) to explicitly refer to the agent’s belief within the language. Given a set \( A \) of formulas (the initial beliefs), a set \( T \) is an expansion of \( A \) if it coincides with the deductive closure of the set \( A \cup \{ L \varphi \mid \varphi \in T \} \cup \{ \neg L \varphi \mid \varphi \notin T \} \). In words, a set \( T \) is an expansion if it equals what can be derived using the initial beliefs \( A \) and positive and negative introspection with respect to \( T \) itself. It was later discovered that this definition of extensions allows unfounded, self-justifying beliefs. Such beliefs are however not always desirable when representing the knowledge of agents.

2.4 The Logic of GK

The language of GK proposed by Lin and Shoham [21] is a modal propositional language with two modal operators, \( K \), for knowledge, and \( A \), for assumption. GK formulas are propositional formulas with \( K \) and \( A \). A GK theory is a set of GK formulas.

GK is a nonmonotonic logic, and its semantics is defined using the standard Kripke possible world interpretations. Informally speaking, a GK model is a Kripke interpretation where what is true under \( K \) is minimal and exactly the same as what is true under \( A \). The intuition here is that given a GK formula, one first makes some assumptions (those true under \( A \)), then one minimizes the knowledge thus entailed, and finally checks to make sure that the initial assumption is justified in the sense that the minimal knowledge is the same as the initial assumption.

Formally, a Kripke interpretation \( M \) is a tuple \( (W, \pi, R_K, R_A, s) \), where \( W \) is a nonempty set of possible worlds, \( \pi \) a function that maps a possible world to an interpretation, \( R_K \) and \( R_A \) binary relations over \( W \) representing the accessibility relations for \( K \) and \( A \), respectively, and \( s \in W \), called the actual world of \( M \). The satisfaction relation \( \models \) between a Kripke interpretation \( M = (W, \pi, R_K, R_A, s) \) and a GK formula \( F \) is defined in a standard way:

\[
\begin{align*}
M & \not\models \bot, \\
M & \models p \text{ iff } p \in \pi(s), \text{ where } p \text{ is an atom,} \\
M & \models \neg F \text{ iff } M \not\models F, \\
M & \models F \land G \text{ iff } M \models F \text{ and } M \models G, \\
M & \models F \lor G \text{ iff } M \models F \text{ or } M \models G, \\
M & \models K^F \text{ iff } (W, \pi, R_K, R_A, \omega) \models F \text{ for any } \omega \in W, \text{ such that } (s, \omega) \in R_K, \\
M & \models A^F \text{ iff } (W, \pi, R_K, R_A, \omega) \models F \text{ for any } \omega \in W, \text{ such that } (s, \omega) \in R_A.
\end{align*}
\]

Note that for any \( \omega \in W \), \( \pi(\omega) \) is an interpretation. We say that a Kripke interpretation \( M \) is a model of a GK formula \( F \) if \( M \) satisfies \( F \), \( M \) is a model of a GK theory \( T \) if \( M \) satisfies every GK formula in \( T \). In the following, given a Kripke interpretation \( M \), we let

\[
K(M) = \{ \phi \mid \phi \text{ is a propositional formula and } M \models K^\phi \}, \\
A(M) = \{ \phi \mid \phi \text{ is a propositional formula and } M \models A^\phi \}.
\]

Notice that \( K(M) \) and \( A(M) \) are always closed under classical logical entailment, that is, they are propositional theories.

Given a GK formula \( T \), a Kripke interpretation \( M \) is a minimal model of \( T \) if \( M \) is a model of \( T \) and there does not exist another model \( M_1 \) of \( T \) such that \( A(M_1) = A(M) \) and \( K(M_1) \subsetneq K(M) \). We say that \( M \) is a GK model of \( T \) if \( M \) is a minimal model of \( T \) and \( K(M) = A(M) \).

In this paper, we consider only GK formulas that do not contain nested occurrences of modal operators. Specifically, an \( A \)-atom is a formula of the form \( A_0 \) and a \( K \)-atom is a formula of the form \( K^\phi \), where \( \phi \) is a propositional formula. A GK formula is called a pure GK formula if it is formed from \( A \)-atoms, \( K \)-atoms and propositional connectives. Similarly, a pure GK theory is a set of pure GK formulas. Given a pure GK formula \( F \), we denote

\[
\text{Atom}_K(F) = \{ \phi \mid K^\phi \text{ is a } K \text{-atom occurring in } F \}, \\
\text{Atom}_A(F) = \{ \phi \mid A^\phi \text{ is an } A \text{-atom occurring in } F \}.
\]

For a pure GK theory \( T \), we use \( \text{Atom}_K(T) = \bigcup_{F \in T} \text{Atom}_K(F) \) and \( \text{Atom}_A(T) = \bigcup_{F \in T} \text{Atom}_A(F) \) to denote their modal atoms.

So far, the applications of the logic of GK only ever use pure GK formulas. We now present some embeddings of well-known nonmonotonic knowledge representation languages into the logic of GK.

Default logic (A propositional) default theory \( \Delta = (W, D) \) (under extension semantics) is translated into pure GK formulas in the following way: (1) Translate each \( \phi \in W \to K^\phi \) (2) Translate each \( \phi \leftarrow \psi_1, \ldots, \psi_n / \varphi \in D \to K^\phi \land \neg A \neg \psi_1 \land \cdots \land \neg A \neg \psi_n \supseteq K^\varphi \). For the weak extension semantics, a default \( \phi \leftarrow \psi_1, \ldots, \psi_n / \varphi \in D \) is translated to \( A^\phi \land \neg A \neg \psi_1 \land \cdots \land \neg A \neg \psi_n \supseteq K^\varphi \).
Logic of universal causation

A UCL formula $F$ is translated to the pure logic of GK by replacing every occurrence of $C$ by $K$, adding $A$ before each atom which is not in the range of $C$ in $F$, and adding $A p \lor \neg A p$ for each atom $p$. For example, if $F$ is $(p \land \neg q) \supset C(p \land \neg q)$ and $Atom = \{ p, q \}$, then the corresponding pure GK formula is $\neg (Ap \land \neg Aq) \supset K(p \land \neg q)$.

Disjunctive logic programs

Similarly, a disjunctive LP rule $p_1 \lor \cdots \lor p_n \leftarrow p_{n+1}, \ldots, p_{m}, \neg p_{m+1}, \ldots, \neg p_m$, where $p_i$'s correspond to the following pure GK formula: $Kp_{n+1} \land \cdots \land Kp_{m} \land \neg A p_{n+1} \land \cdots \land \neg A p_{m} \land Kp_1 \lor \cdots \lor Kp_n$.

Main Result: From Pure GK to Disjunctive ASP

Before presenting the translation, we introduce some notations. Let $F$ be a pure GK formula, we use $tr_{p}(F)$ to denote the propositional formula obtained from $F$ by replacing each occurrence of a $K$-atom $K \phi$ by $k_\phi$ and each occurrence of an $A$-atom $A \psi$ by $a_\psi$, where $k_\phi$ and $a_\psi$ are new atoms with respect to $\phi$ and $\psi$ respectively. For a pure GK theory $T$, we define $tr_{p}(T) = \bigwedge_{F \in T} tr_{p}(F)$. To illustrate these and the definitions that follow, we use a running example.

Example 1 (Normal Reiter default)

Consider the pure GK theory $\{ F \}$ with $F = \neg A \neg p \supset Kp$ corresponding to the default $T : p \leftarrow 1$. Then $tr_{p}(\{ F \}) = \neg a_\neg p \lor Kp$, where $a_\neg p$ and $k_p$ are new atoms.

Here we introduce a set of new atoms $k_\phi$ and $a_\psi$ for each formula $\phi \in Atom_{GK}(T)$ and $\psi \in Atom_{GK}(T)$. Intuitively, the new atom $k_\phi$ (resp. $a_\psi$) will be used to encode containment of the formula $\phi$ in $K(M)$ (resp. $A(M)$) of a GK model $M$ for $T$.

Given a propositional formula $\phi$ and an atom $a$, we use $\phi^a$ to denote the propositional formula obtained from $\phi$ by replacing each occurrence of an atom $p$ with a new atom $p^a$ with respect to $a$. These formulas and new atoms will later be used in our main translation to perform the minimality check of the logic of GK's semantics.

We now stepwise walk our way towards the main result. We start with a result that relates a pure GK theory to a propositional formula that will later reappear in our main translation.

Proposition 1

Let $T$ be a pure GK theory. A Kripke interpretation $M$ is a model of $T$ if and only if there exists a model $I^*$ of the propositional formula $\Phi_T = tr_p(T) \land \varphi_{\text{end}} \land K \varphi_{\text{end}} \land \varphi_{\text{end}}$ where

- $\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{GK}(T)} (k_\phi \supset \phi^k) \land \bigwedge_{\phi \in Atom_{A}(T)} (a_\phi \supset \phi^a)$

- $K \varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{GK}(T)} \neg k_{\neg \phi} \lor \bigwedge_{\phi \in Atom_{GK}(T)} (k_\phi \supset \phi^{k_\phi})$

- $\varphi_{\text{end}} = \bigwedge_{\psi \in Atom_{A}(T)} \neg a_\psi \lor \bigwedge_{\phi \in Atom_{A}(T)} (a_\phi \supset \phi^{a_\phi})$

such that

- $K(M) \cap Atom_{GK}(T) = \{ \phi \mid \phi \in Atom_{GK}(T) \text{ and } I^* \models k_\phi \}$

- $A(M) \cap Atom_{A}(T) = \{ \phi \mid \phi \in Atom_{A}(T) \text{ and } I^* \models a_\phi \}$

The proposition examines the relationship between models of a pure GK theory and particular models of the propositional formula $\Phi_T$. The first conjunct $tr_p(T)$ of the formula $\Phi_T$ indicates that the $k$-atoms and $a$-atoms in it can be interpreted in accordance with $K(M)$ and $A(M)$ such that $I^* \models tr_p(T)$ iff $M$ is a model of $T$.

The soundness formula $\varphi_{\text{end}}$ achieves that the sets $\{ \phi \mid \phi \in Atom_{GK}(T) \text{ and } I^* \models k_\phi \}$ are consistent. The witness formulas $\varphi_{\text{end}}$ and $\varphi_{\text{end}}$ indicate that, if $I^* \models k_\phi$ for some $\psi \in Atom_{GK}(T)$ (resp. $\psi \in Atom_{A}(T)$) then there exists a model $I^*$ of $K(M)$ (resp. $A(M)$) such that $I^* \models \neg \psi$, where $I^*$ is explicitly indicated by newly introduced $p^{k_\phi}$ (resp. $p^{a_\phi}$) atoms. So intuitively, if a formula is not known (or not assumed), then there must be a witness for that. This condition is necessary; for instance, if the set $\{ k_{p}, k_{q} \}$ satisfies the formulas $k_{p,q} \lor k_{p}$ and $k_{p,q} \lor k_{q}$, however, since $K(M)$ is a theory there does not exist a Kripke interpretation $M$ such that $p \in K(M)$, $q \in K(M)$ and $p \land q \notin K(M)$.

Example 1 (Continued)

The formula $\Phi_{p,j}$ is given by:

$tr_p(F) = \neg a_\neg p \lor Kp$

$\varphi_{\text{end}} = (k_p \lor p^k) \land (a_\neg p \lor \neg p^a)$

$K \varphi_{\text{end}} = \neg k_p \lor (p^{k_p} \lor k_p)$

$\varphi_{\text{end}} = \neg a_\neg p \lor (\neg p^{a_\neg p} \lor (a_\neg p \lor \neg p^{a_\neg p}))$

While Proposition 1 aligns Kripke models and propositional models of the translation, there is yet no mention of GK's typical minimization step. This is the task of the next result, which extends the above relationship to GK models.

Proposition 2

Let $T$ be a pure GK theory. A Kripke interpretation $M$ is a GK model of $T$ if and only if there exists a model $I^*$ of the propositional formula $\Phi_T$ such that

- $K(M) = A(M) = Th(\{ \phi \mid \phi \in Atom_{GK}(T) \text{ and } I^* \models k_\phi \})$

- for each $\psi \in Atom_{GK}(T)$,

$\varphi_{\text{end}}$ does not exist another model $I^*$ such that

- $\neg \varphi_{\text{end}} \land \varphi_{\text{end}}$ with

$\varphi_{\text{end}}$ is clearly the intended reading of our running example is that there is no reason to assume that $p$ is false, and the default lets us conclude that we know $p$. This is testified by the partial interpretation $I^* = \{ \neg a_\neg p, k_p, p^{k_p}, p^{a_\neg p} \}$ where the remaining atoms are not relevant. It is easy to verify that $I^*$ is a model for $\Phi_{p,j}$ and there is no model $I^*$ with the properties above. Now $k_p \in I^*$ shows that $p$ is known in the corresponding GK model.

In Proposition 2, we only need to consider a Kripke interpretation $M$ such that $A(M) \cup K(M)$ is consistent. This means that formula $\Phi_T$ can be modified to $\Psi_T = tr_{p}(T) \land \varphi_{\text{end}} \land \varphi_{\text{end}}$ with

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{GK}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{A}(T)} (a_\phi \lor \phi^{a_\phi})$

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{A}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{GK}(T)} (a_\phi \lor \phi^{a_\phi})$

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{A}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{GK}(T)} (a_\phi \lor \phi^{a_\phi})$

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{A}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{GK}(T)} (a_\phi \lor \phi^{a_\phi})$

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{A}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{GK}(T)} (a_\phi \lor \phi^{a_\phi})$

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{A}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{GK}(T)} (a_\phi \lor \phi^{a_\phi})$

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{A}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{GK}(T)} (a_\phi \lor \phi^{a_\phi})$

$\varphi_{\text{end}} = \bigwedge_{\phi \in Atom_{A}(T)} (k_\phi \lor \phi^{k_\phi}) \land \bigwedge_{\phi \in Atom_{GK}(T)} (a_\phi \lor \phi^{a_\phi})$
\( \Phi^k_T = \neg p^k \land \bigwedge_{\phi \in \text{Atom}_T} (k \vdash \phi^k) \land \bigwedge_{\phi \in \text{Atom}_T} (a \vdash \phi^a) \)

\( \Phi^a_T = \neg \psi^a \land \bigwedge_{\phi \in \text{Atom}_T} (k \vdash \phi^a) \land \bigwedge_{\phi \in \text{Atom}_T} (a \vdash \phi^a) \)

So the soundness formula \( \varphi_{\text{out}} \) actually becomes easier, since soundness of knowledge and assumptions is enforced for one and the same vocabulary (the one from the original theory). The witness formulas become somewhat more complicated, as the witnesses have to respect both the knowledge as well as the assumptions of the theory. This is best explained by consulting our running example again.

**Example 1 (Continued)** While \( F \)'s propositionalization \( \text{tr}_p(\{F\}) \) stays the same, the soundness and witness formulas change in the step from formula \( \Phi(p) \) to formula \( \Psi(p) \). We only show the first conjunct of the witness formula \( \varphi_{\text{out}} \), which is given by

\[-k_p \vdash \left( -p^b \land (k \vdash p^b) \land (a \vdash \neg p^b) \right)\]

Intuitively, the formula expresses that whenever \( p \) is not known, then there must be a witness, that is, an interpretation where \( p \) is false. Since the witnessing interpretations could in principle be distinct for each \( K \)-atom, they have to be indexed by the respective \( K \)-atom they refer to, as in \( p^b \). Of course, the witnesses have to obey all that is known and assumed, which is guaranteed in the last two conjuncts.

Using this new formula, the result of Proposition 2 can be restated.

**Proposition 3** Let \( T \) be a pure \( GK \) theory. A Kripke interpretation \( M \) is a \( GK \) model of \( T \) if and only if there exists a model \( I^* \) of the propositional formula \( \Psi_T \) such that

- \( K(M) = A(M) = \text{Th}(\{ \phi \mid \phi \in \text{Atom}_T \text{ and } I^* \models k \}) \);
- for each \( \psi \in \text{Atom}_T \),
  
  \( I^* \models a \) then \( \psi \in \text{Th}(\{ \phi \mid \phi \in \text{Atom}_T \text{ and } I^* \models k \}) \);
- there does not exist another model \( I'' \) of \( \Psi_T \) such that
  - \( I'' \cap \{ a \mid \phi \in \text{Atom}_T \} = I^* \cap \{ a \mid \phi \in \text{Atom}_T \} \),
  - \( I'' \cap \{ k \mid \phi \in \text{Atom}_T \} \not\subseteq I^* \cap \{ k \mid \phi \in \text{Atom}_T \} \).

We are now ready for our main result, translating a pure \( GK \) theory to a disjunctive logic program. First, we introduce some notations. Let \( T \) be a pure \( GK \) theory, we use \( \text{tr}_{\text{w}}(T) \) to denote the nested expressions obtained from \( \Psi_T \) by first converting it to negation normal form\(^3\), then replacing "\( \land \)" by "\( -\)" and "\( \lor \)" by "\( \lor \)". A propositional formula \( \phi \) can be equivalently translated to conjunctive normal form (involving at most linear blowup)

\[(p_1 \lor \cdots \lor p_n) \land \cdots \land (q_1 \lor \cdots \lor q_n)\]

where \( p \)'s and \( q \)'s are atoms; we use \( \text{tr}_c(\phi) \) to denote the set of rules

\[p_1 \cdots p_i \leftarrow p_{i+1}, \ldots, p_m, \quad \cdots \quad q_1 \cdots q_k \leftarrow q_{k+1}, \ldots, q_n\]

We use \( \hat{\phi} \) to denote the propositional formula obtained from \( \phi \) by replacing each occurrence of an atom \( p \) by a new atom \( \hat{p} \).

We use \( T'' \) to denote the propositional formula obtained from the formula \( \Phi_T \) by replacing each occurrence of an atom \( p \) (except atoms in \( \{ a \mid \phi \in \text{Atom}_T \} \)) by a new atom \( p' \). Intuitively, each atom that is not an \( a \)-atom is replaced by a new atom.

Notice that \( \text{tr}_{\text{w}}(T) \) is obtained from \( \Phi_T \) while \( T'' \) is obtained from \( \Psi_T \). Intuitively, by Proposition 3, \( \text{tr}_{\text{w}}(T) \) is used to restrict interpretations for introduced \( k \)-atoms and \( a \)-atoms so that these interpretations serve as candidates for \( GK \) models, and by Proposition 1, \( T'' \) constructs possible models of the \( GK \) theory which are later used to test whether these models prevent the candidate to be a \( GK \) model.

Inspired by the linear translation from parallel circumscription into disjunctive logic programs in [13], we have the following theorem.

**Theorem 1** Let \( T \) be a pure \( GK \) theory. A Kripke interpretation \( M \) is a \( GK \) model of \( T \) if and only if there exists an answer set \( S \) of the logic program \( \text{tr}_p(T) \):

1. \( \perp \leftarrow \text{not} \text{tr}_{\text{w}}(T) \)
2. \( p' \leftarrow \text{not} \text{tr}_{\text{w}}(T) \) (for each atom \( p' \) occurring in \( \text{tr}_{\text{w}}(T) \))
3. \( u; A \leftarrow B \) (for each rule \( A \leftarrow B \) in \( \text{tr}_{\text{c}}(T) \))
4. \( w; c_{\text{wh}} \leftarrow T \) (for each \( \{ \phi_1, \ldots, \phi_n \} = \text{Atom}_T \))
5. \( u \leftarrow c_{\text{wh}}, \text{not} \text{tr}_{\text{w}}(T) \) (for each \( \phi \in \text{Atom}_T \))
6. \( u \leftarrow k, \text{not} \text{tr}_{\text{w}}(T) \) (for each \( \phi \in \text{Atom}_T \))
7. \( w; c_{\text{wh}}, k \leftarrow \text{not} \text{tr}_{\text{w}}(T) \) (for each \( \phi \in \text{Atom}_T \))
8. \( p' \leftarrow u \) (for each new atom \( p' \) in \( \text{tr}_{\text{c}}(T) \))
9. \( c_{\text{wh}} \leftarrow u \) (for each \( \phi \in \text{Atom}_T \))
10. \( \perp \leftarrow \text{not} u \)
11. \( u; A \leftarrow B \)

(13) for each rule \( A \leftarrow B \) in the \( \text{tr}_{\text{c}}(\cdot) \) translation of

\[ \bigwedge_{\phi \in \text{Atom}_T} (k \vdash \phi) \land \bigwedge_{\phi \in \text{Atom}_T} (a \vdash \phi) \]

\( \hat{\phi} \leftarrow v \)

(14) for each atom \( \hat{p} \) except \( k \)-atoms and \( a \)-atoms in \( \text{tr}_{\text{w}}(\cdot) \) of

\[ \bigwedge_{\phi \in \text{Atom}_T} (k \vdash \phi) \land \bigwedge_{\phi \in \text{Atom}_T} (a \vdash \phi) \]

\( \perp \leftarrow \text{not} v \) where \( u, v, \) and \( c_{\text{wh}} \) (for each \( \phi \in \text{Atom}_T \)) are new atoms, such that

\[ K(M) = A(M) = \text{Th}(\{ \phi \mid \phi \in \text{Atom}_T \text{ and } k \in S \}) \]

The intuition behind the construction is as follows:

- (1) and (2) in \( \text{tr}_p(T) \): \( I^* \) is a model of the formula \( \Psi_T \).
- (3) to (8): if there exists a model \( I'' \) of the formula \( \Phi_T \) with
  - \( \perp \leftarrow \text{not} \text{tr}_{\text{w}}(T) \)
  - \( I'' \cap \{ k \mid \phi \in \text{Atom}_T \} \not\subseteq I^* \cap \{ k \mid \phi \in \text{Atom}_T \} \),
  then there exists a set \( S^* \) constructed from new atoms in \( \text{tr}_{\text{c}}(T) \) (which is a copy of the formula \( \Phi_T \) with same \( a_0 \) for each \( \phi \in \text{Atom}_T \)) and \( c_{\text{wh}} \) for some \( \phi \in \text{Atom}_T \) such that \( S^* \) satisfies rules (3) to (8) and \( u \notin S^* \).
- (9) and (10): if there is such a set \( S^* \) then it is the least set containing \( u, v \)'s and \( c_{\text{wh}} \).
- (11): such a set \( S^* \) should not exist. (See item 3 in Proposition 3.)
- (12) and (13): if there exists a model of the propositional formula

\[ \bigwedge_{\phi \in \text{Atom}_T} (k \vdash \phi) \land \bigwedge_{\phi \in \text{Atom}_T} (a \vdash \phi) \]

then \( v \) should not occur in the minimal model of the program.

\(^3\) A propositional formula is in Negation Normal Form (NNF) if negation occurs only immediately above atoms, and \( \perp, \top, \neg, \land, \lor \) are the only allowed connectives.
Computational complexity We have seen in Section 2.4 that disjunctive logic programs can be modularly and equivalently translated into pure formulas of the logic of GK. Conversely, Theorem 1 shows that pure GK formulas can be equivalently translated into disjunctive logic programs. Eiter and Gottlob showed that the problem of deciding whether a disjunctive logic program has an answer set is \( \Sigma^P_2 \)-complete [8]. In combination, these results yield the following straightforward complexity result for the satisfiability of pure GK.

**Proposition 4** Let \( T \) be a pure GK theory. The problem of deciding whether \( T \) has a GK model is \( \Sigma^P_2 \)-complete.

4 Implementation

We have implemented the translation of Theorem 1 into a working prototype \( \text{gk2dlp} \). The program is written in Prolog and uses the disjunctive ASP solver claspD-2 [10], which was ranked first place in the 2013 ASP competition (http://www.mat.unical.it/iansi/storage/aspc-comp-2013-lpnmttalk.pdf). When computing answer sets of the translated logic programs, we use claspD-2’s “--project” option to project out all atoms but \( K \)-atoms, since these suffice to reconstruct GK models.

Our prototype is the first implementation of the (pure) logic of GK to date. The restriction to pure formulas seems harmless since all known applications of the logic of GK use only pure formulas. We remark that \( \text{gk2dlp} \) implements default and autoepistemic logics such that input and target language are of the same complexity.

**Evaluation** To have a scalable problem domain and inspired by dl2asp [4], we chose the fair division problem [2] for experimental evaluation. An instance of the fair division problem consists of a set of agents, a set of goods, and for each agent a set of constraints that intuitively express which sets of goods the agent is willing to accept. A solution is then an assignment of goods to agents that is a partition of all goods and satisfies all agents’ constraints. Bouvier & Lang [2] showed that the problem can be naturally encoded in default logic.

We created random instances of the fair division problem with increasing numbers of agents and goods. We then applied the translation of [2]; furthermore the translation from default logic into the logic of GK, then invoked \( \text{gk2dlp} \) to produce logic programs and finally used gringo 3.0.3 and claspD version 2 (revision 6814) to compute all answer sets of these programs, thus all extensions of the original default theory corresponding to all solutions of the problem instance. The experiments were conducted on a Lenovo laptop with an Intel Core i3 processor with 4 cores and 4GB of RAM running Ubuntu 12.04. We recorded the size of the default theory, the size of the translated logic program, the translation time and the solving time, as well as the number of solutions obtained. We started out with 2 agents and 2 goods, and stepwise increased these numbers towards 6. For each combination in \( \{a, g\} \subseteq \{2, \ldots, 6\} \times \{2, \ldots, 6\} \), we tested 20 randomly generated instances. Random generation here means that we create agents’ preferences by iteratively drawing random subsets of goods to add to an agent’s acceptable subsets with probability \( P \), where \( P \) is initialized with 1 and discounted by the factor \( 1 - \frac{1}{\eta} \) for each subset that has been drawn.

Below we show two scatter plots, where each point represents a single problem instance. The first plot depicts the size increase of the translation of this paper. The size of the GK formula of a given default theory is roughly the same as that of the default theory. As the plot shows, the increase in size from default theory to logic program is polynomial, albeit with a low exponent. The second plot shows the solving time in relation to the size of the default theory, where the time axis is logarithmic. We can see that the runtime behavior of \( \text{gk2dlp} \) is satisfactory, although not competitive with that of dl2asp [4]. However, a direct comparison of the two systems is problematic since [4] do not describe how they create random instances of the fair division problem, and more importantly dl2asp is especially engineered for default logic, while \( \text{gk2dlp} \) is a more general system.

![Figure 1. Size increase from default theory to disjunctive logic program.](image1)

![Figure 2. Solving time (log scale) with respect to default theory size.](image2)
al. [1] introduced a method for default reasoning in action theories, that is, an approach to the question what normally holds in a dynamic domain. Our translation yields an implementation of their approach, something that they stated as future work and later achieved to a limited extent (for a restricted sublanguage of their framework [34]).

Related work There are few approaches that implement as broad a range of propositional nonmonotonic knowledge representation languages as gk2asp. Two notable exceptions are the works of Junker and Konolige [15], who implemented both autoepistemic and default logics by translating them to truth maintenance systems; and Niemelä [29], who provides a decision procedure for autoepistemic logic which also incorporates extension semantics for default logics. Other approaches are restricted to specific languages. For default logic, the recent system dl2asp [4] translates default theories to normal (non-disjunctive) logic programs; the translation figures out all implication relations between formulas occurring in the default theory, just as [15] did. The authors of dl2asp [4] already observed that default logic and disjunctive logic programs are of the same complexity; they even stated the search for a polynomial translation from the former to the latter (that we achieved in this paper) as future work. Gadel [28] uses a genetic algorithm to compute extensions of a default theory; likewise the system DeReS [5] is not translation-based but directly searches for extensions; similarly the XRay system [33] natively implements local query-answering in default logics. Risch and Schwind [32] describe a tableau-based algorithm for computing all extensions of general default theories, but do not report runtimes for their Prolog-based implementation. For autoepistemic logic, Marek and Truszczynski [23] investigate sceptical reasoning with respect to Moore’s expansion semantics.

5 Discussion

We have presented the first translation of pure formulas of the logic of GK to disjunctive answer set programming. Among other things, this directly leads to implementations of Turner’s logic of universal causation as well as implementations of default and autoepistemic logics under different semantics. We have prototypically implemented the translation and experimentally analysed its performance, which we found to be satisfactory given the system’s generality.

In the future, we plan to integrate further nonmonotonic reasoning formalisms. This is more or less straightforward due to the generality of this work: to implement a language, it suffices to provide a translation into pure formulas of GK, then Theorem 1 of this paper does the rest. A particular formalism we want to look at is Lakemeyer and Levesque’s logic of only-knowing [17]. We also plan to study the approaches mentioned as applications in the previous section to try out our translation and implementation on agent-oriented AI problems.

References