Iterated Belief Change, Computationally

Kai Sauerwald, Christoph Beierle
FernUniversität in Hagen, Knowledge-Based Systems, 58084 Hagen, Germany
{kai.sauerwald, christoph.beierle}@fernuni-hagen.de

Abstract
This paper considers belief change in the Darwiche-Pearl framework. We demonstrate that iterative belief revision is Turing complete by showing how revision operators over ranking functions can simulate every Turing machine. Our result holds even under the condition that the broadly accepted Darwiche-Pearl postulates for iterated revision hold.

1 Introduction
The question of how an agent should change her belief according to new information is a central task for agents and AI systems. The belief change community developed a very fruitful unique research approach to this subject. The approach is top-down, by considering postulates, while being at the same time agent-centric, by taking a subjective perspective on the phenomenon of belief change.

Iterated belief change considers relations between a change and changes that happen priorly or posteriorly. In this area, changes of epistemic states are considered, which may contain “meta-information” in addition to the plain beliefs. The framework by Darwiche and Pearl (1997) deals with the aspect of meta-information elegantly by considering belief change over abstract sets of epistemic states. The computation complexity in specific sets of epistemic states was investigated for iterated belief revision (Liberatore 1997; Schwind et al. 2020), however, investigations on the general computability are missing.

In this paper, we focus on the computational aspect of iterated belief change. We quantify its computational power by providing corresponding notions and a construction that demonstrate that iterated belief change is Turing complete, even when the prominent iteration principles of Darwiche and Pearl (1997) are satisfied. We will therefore consider iterated revision in the Darwiche-Pearl framework and instantiate the set of epistemic states by ranking functions.

2 Formal Background
Let $\Sigma = \{a, b, c, \ldots\}$ be a propositional signature (non empty finite set of propositional variables) and $L$ a propositional language over $\Sigma$. The set of propositional interpretations is denoted by $\Omega$. Propositional entailment is denoted by $\models$, the set of models of $\alpha$ with $[\alpha]$, and $L \subseteq L$ is deductively closed if $L = \{\beta \mid L \models \beta\}$. For a set of worlds $\Omega' \subseteq \Omega$ and a total preorder $\preceq \subseteq \Omega \times \Omega$ (total, reflexive and transitive relation), we denote with $\min(\Omega', \preceq) = \{\omega \in \Omega' \mid \omega \preceq \omega' \text{ for all } \omega' \in \Omega'\}$ the set of all minimal worlds of $\Omega'$ with respect to $\preceq$. For a total preorder $\preceq$, we say $x \prec y$ if $x \preceq y$ and $y \not\preceq x$. With $\preceq$ we denote the usual ordering on the integers and with $<$ we denote the strict part of $\preceq$.

3 Iterated Belief Revision in the Darwiche-Pearl Framework
AGM theory, by Alchourrón, Gärdenfors and Makinson (1985), deals with the dynamics of beliefs in the context of belief sets, i.e., deductively closed sets of propositions.

The area of iterated belief change abstracts from belief sets to epistemic states, in which the agent maintains necessary information for all used belief change operators. The most prominent approach for iterated belief change is the framework by Darwiche and Pearl (1997), which does leave open what an epistemic state is, but assumes that for every epistemic state $\Psi$ we can obtain the set of plausible sentences $\text{Bel}(\Psi) \subseteq L$ of $\Psi$, which is deductively closed.

While the Darwiche-Pearl framework is very successful, an ontological problem of belief change itself appears in their framework: when describing change operators over epistemic states, we have to specify on which epistemic states the changes happen, an aspect which was not explicitly treated by Darwiche and Pearl and rediscovered several times (Friedman and Halpern 1996; Aravanis, Peppas, and Williams 2019; Schwind and Konieczny 2020).

We solve this problem by considering an abstract set of epistemic states $\mathcal{E}$, and assume in this paper that $\mathcal{E}$ is a countable infinite set and that for every belief set $L \subseteq L$ there exists infinite many epistemic states $\Psi \in \mathcal{E}$ with $\text{Bel}(\Psi) = L$. We write $\Psi \models \alpha$ if $\alpha \in \text{Bel}(\Psi)$, and we define $[\Psi] = [\text{Bel}(\Psi)]$.

Definition 1. A belief change operator (for $\mathcal{E}$ and $L$) is a function $\circ : \mathcal{E} \times L \to \mathcal{E}$.

As we do not specify in detail what an epistemic state in $\mathcal{E}$ is, one has to identify the additional information for the elements of $\Psi$ for realizing certain operators. Formally, we do this by assigning additional information (similar as Katsuno and Mendelzon, 1992) to epistemic states.

Definition 2. A function $\Psi \mapsto \preceq_\Psi$ which assigns to each epistemic states $\Psi \in \mathcal{E}$ a preorder $\preceq_\Psi$ over $\Omega$ is called a faithful assignment if $[\Psi] \neq \emptyset$ implies $[\Psi] = \min(\Omega, \preceq_\Psi)$. 

410
AGM revision is characterized by selecting minimal models in the respective assigned total preorders.

Proposition 3 (AGM Revision for Epistemic States (Darwiche and Pearl 1997)). A belief change operator * is an AGM revision operator for epistemic states if there is a faithful assignment $\Psi \mapsto \preceq\Psi$, which assigns to $\Psi$ a total preorder $\preceq\Psi$ such that:

$$\|\Psi \circ \alpha\| = \min(\|\alpha\|, \preceq\Psi)$$ (1)

Darwiche and Pearl (1997) proposed the following postulates:

1. (DP1) if $\beta \models \alpha$, then $\Bel(\Psi \circ \alpha \circ \beta) = \Bel(\Psi \circ \beta)$
2. (DP2) if $\beta \models -\alpha$, then $\Bel(\Psi \circ \alpha \circ \beta) = \Bel(\Psi \circ \beta)$
3. (DP3) if $\Psi \circ \beta \models \alpha$, then $(\Psi \circ \alpha) \circ \beta \models \alpha$
4. (DP4) if $\Psi \circ \beta \not\models -\alpha$, then $(\Psi \circ \alpha) \circ \beta \not\models -\alpha$

It is well-known that these operators can be characterized in the semantic framework of total preorders.

Proposition 4 (Darwiche and Pearl (1997)). Let * be an AGM revision operator for epistemic states. Then * satisfies (DP1)–(DP4) if and only there exists a faithful assignment $\Psi \mapsto \preceq\Psi$ such that $\preceq\Psi$ is a total preorder, Equation (1) holds, and the following postulates are satisfied:

1. (CR1) if $\omega_1, \omega_2 \in [\alpha]$, then $\omega_1 \preceq\Psi \omega_2 \Rightarrow \omega_1 \preceq\Psi \omega_2$
2. (CR2) if $\omega_1, \omega_2 \in [-\alpha]$, then $\omega_1 \preceq\Psi \omega_2 \Rightarrow \omega_1 \preceq\Psi \omega_2$
3. (CR3) if $\omega_1 \in [\alpha], \omega_2 \in [-\alpha]$, then $\omega_1 \preceq\Psi \omega_2 \Rightarrow \omega_1 \preceq\Psi \omega_2$
4. (CR4) if $\omega_1 \in [\alpha], \omega_2 \in [-\alpha]$, then $\omega_1 \preceq\Psi \omega_2 \Rightarrow \omega_1 \preceq\Psi \omega_2$

The postulates (DP1)–(DP4) are accepted today as very important iteration principles for revision.

4 Turing Completeness

In this section, we will show that iterated belief revision is inherently computationally powerful, in particular, we will see that belief change operators can simulate Turing machines.

4.1 Computability Theory

At first, we recall basic concepts of computability theory and formal language theory. In the following, $\Gamma = \{\varepsilon, \sigma, \ldots\}$ will denote a finite (non-empty) alphabet and $\Gamma^* = \{\varepsilon, \sigma, \varepsilon\sigma, \ldots\}$ denotes the set of all finite words over $\Gamma$, where $\varepsilon$ is the Kleene star operator and $\varepsilon$ denotes the empty word.

A (deterministic) Turing machine is a tuple $T = (Q, \Gamma, \delta, q_0)$ where $Q = \{q_0, \ldots, q_n\}$ is a finite set of states where $q_0 \in Q$ is the starting state, the set $\Gamma$ is the finite alphabet of tape symbols, and $\delta : Q \times \Gamma \rightarrow Q \times \{\leftarrow, \downarrow, \rightarrow\}$ is the transition function of $T$. The state $q_{halt} := q_n$ is used a special state, indicating that the machine $T$ has stopped. We describe the position on the tape of $T$ as triple $\langle t_1, t_c, t_r \rangle \in \Gamma^* \times \Gamma \times \Gamma^*$, where the middle symbol is used to denote the current scanning position. A configuration of $T$ is a tuple $\langle q, \langle t_1, t_c, t_r \rangle \rangle \in Q \times \Gamma^* \times \Gamma$. The start configuration for $T$ by input $\gamma = \sigma_1 \sigma_2 \cdots \sigma_n \in \Gamma^*$ is $\langle q_0, \varepsilon, \sigma_1, \varepsilon, \sigma_2, \ldots, \sigma_n \rangle$. The run $r$ of $T$ by input $\gamma \in \Gamma^*$ is defined in the usual way as a (possibly infinite) sequence of configurations, starting in the corresponding start configuration by input $\gamma \in \Gamma^*$, and all subsequent configurations are determined by $\delta$ in the usual way. A configuration $\langle q, \langle t_1, t_c, t_r \rangle \rangle$

with $q = \theta_{halt}$ is a halting configuration, and we require that after a halting configuration there are no subsequent configurations in a run. If the run $r$ starting in $\gamma$ ends in a halting configuration $\langle q, \langle t_1, t_c, t_r \rangle \rangle$ we say the word $t_1 t_c t_r$ is the output of $\gamma$ by $T$, if $r$ does not halt, then $\infty$ is the output of $\gamma$ by $T$.

Definition 5. A partial function $f : \Gamma^* \rightarrow \Gamma^*$ is called Turing computable if there is a Turing machine $T = (Q, \Gamma, \delta, q_0)$ such that $f(\gamma)$ is the output of $\gamma$ by $T$ for every $\gamma \in \Gamma^*$. If this is the case, we say that $T$ computes $f$.

4.2 Ranking Function-Based Change Operators

A function $\kappa : \Omega \rightarrow \mathbb{N}_0$ is a ranking function, introduced (in a more general form) by Spohn (1988; 2012), if there exists some $\omega \in \Omega$ such that $\kappa(\omega) = 0$. Let $\{\kappa\} = \{\omega \in \Omega \mid \kappa(\omega) = 0\}$ and $\Bel(\kappa) = \{\alpha \in \mathcal{L} \mid \kappa \subseteq \alpha\}$. Every $\kappa$ gives rise to a total preorder $\preceq\kappa$, given by $\omega_1 \preceq\kappa \omega_2$ if $\kappa(\omega_1) \leq \kappa(\omega_2)$. For our main theorem we make use of special change operators, which can be characterized by information in form of a ranking function.

Definition 6. A belief change operator $\circ$ is called ranking function-based if there is an assignment $\Psi \mapsto (\kappa_\Psi, b_\Psi)$ such that the following is satisfied:

(RFA0) if $[\Psi] = \emptyset$, then $b_\Psi = \bot$; otherwise $b_\Psi = T$
(RFA1) if $b_\Psi = T$, then $[\Psi] = [\Psi]$
(RFA2) if $\kappa_\Psi = \kappa_\Psi$ and $b_\Psi = b_\Psi$, then $\Psi \circ \alpha = \Phi \circ \alpha$

Note that Definition 6 makes again no assumption about the real form of epistemic states, but guarantees that the behaviour of a belief change operator is representable by an assigned ranking function $\kappa_\Psi$. The following proposition relies heavily on our assumptions about $\mathcal{E}$.

Proposition 7. A belief change operator $\circ$ is an AGM revision operator for epistemic states if and only if $\circ$ is ranking function-based with $\Psi \mapsto (\kappa_\Psi, b_\Psi)$ such that:

$$\|\Psi \circ \alpha\| = \min(\|\alpha\|, \preceq\kappa_\Psi)$$ (2)

Proof. We consider both directions of the proof independently, but due to space reasons we focus only on $\Psi$ with $[\Psi] \neq \emptyset$, and do not consider the $b_\Psi$-part. The “if” direction. Let $\Psi \mapsto \preceq\Psi$ be a faithful assignment which satisfies Equation (1), guaranteed by Proposition 3. With $\Behv(\Psi) = \{\Phi \in \mathcal{E} \mid \preceq\Psi = \preceq_\Psi$ and $\Psi \circ \alpha = \Phi \circ \alpha$ for all $\alpha \in \mathcal{L}\}$ we denote all behavioural equivalent states to $\Psi$. Clearly, $\Phi \in \Behv(\Psi)$ if and only if $\Behv(\Psi) = \Behv(\Psi)$. Note that $\Behv(\Psi)$ is countable and that for each $\preceq\Psi$ there exist countably infinite many ranking functions $\kappa$ with $\preceq\Psi = \preceq_\Psi$. Because of these properties, we can construct $\Psi \mapsto \kappa_\Psi$ by choosing for each different set $\Behv(\Psi)$ a unique ranking function $\kappa_\Psi$ with $\preceq\Psi = \preceq_\Psi$. By construction (RFA1), (RFA2) and Equation (2) are satisfied by $\Psi \mapsto \kappa_\Psi$.

The “only if” direction. Every assignment $\Psi \mapsto \kappa_\Psi$ as given in Definition 6 which satisfies Equation (2) yields a faithful assignment $\Psi \mapsto \preceq_\Psi$ which satisfies Equation (1). Thus, by Proposition 3, existence of $\Psi \mapsto \kappa_\Psi$ as above implies that $\circ$ is an AGM revision operator for epistemic states.

\[\]
Because of Proposition 7, we will focus on ranking function-based change operators and identify epistemic states with ranking functions, and use $\kappa$ as a synonym for $\Psi$ with $[\Psi] \neq \emptyset$.

### 4.3 Simulation Machinery

We will simulate one step of a Turing machine by multiple changes. The simulation uses four types of ranking functions with special meaning in the simulation of $T$:

**CONF** Intended to represent a configuration of $T$.

**PEEK** Intended to check where $T$ is in a halting state.

**TRANS** Describing the change of the state in a transition (before modifying the tape).

**POST** An intermediate state after updating state and tape.

In particular, simulation of a transition to a new configuration will take three steps. As first step we will compute a **TRANS** ranking function where we update the state of $T$ (while saving the old state). Then we update the tape, obtaining a **POST** ranking function. Adjusting some ranks yields a **CONF** ranking function representing the new configuration (see Algorithm 1 for an overview).

The algorithm functions of these types are given over a signature $\Sigma$ with $|\Sigma| = 2$. Consequently, $\Omega$ has four elements, which we denote by $\omega_0, \omega_{halt}, \omega_q, \omega_{pos}$. Each interpretation from $\Omega$ will serve a special purpose (see Figure 1). We use the rank of $\omega_q$ as indicator for the current state of $T$. The interpretation $\omega_{pos}$ will encode the tape position of $T$ using the rank of $\omega_{pos}$. Therefore, remember, that it is well-known that (by a variation of a Gödel numbering) for a fixed $\Gamma$ we can encode positions $(t_i, t_e, t_r)$ into natural numbers. Let $\text{enc} : \Gamma^* \times \Gamma \times \Gamma^* \to \mathbb{N}$ be such an injective encoding function. We use $\omega_{halt}$ to check if the halting state is reached (PEEK) and for saving the prior state of $T$ (TRANS). The interpretation $\omega_q$ has rank 0 in most situations; the purpose is to ensure that we always have a ranking function.

In the following we will describe the four types of ranking functions in more detail:

**CONF.** A **CONF** ranking function $\kappa$ is given if

\[
\begin{aligned}
\kappa(\omega_0) &= 0 & 1 \leq \kappa(\omega_q) &\leq n \\
\kappa(\omega_{halt}) &= n & n < \kappa(\omega_{pos})
\end{aligned}
\]

and $\text{enc}^{-1}(\kappa(\omega_{pos}) - (n + 1)) \neq \emptyset$ holds. For such a ranking function let $\text{conf}(\kappa) = \langle q_n(\omega_0), \text{enc}^{-1}(\kappa(\omega_{pos}) - (n + 1)) \rangle$ denote the represented configuration of $T$. Note that we can provide for each possible configuration a ranking function that represents the configuration. In particular, for a given Turing machine $T$ we denote with $\kappa_T$ a ranking function representing the start configuration for $T$ by input $\gamma \in \Gamma^*$.

**PEEK.** When performing a revision on a **CONF** ranking function by a formula having $\omega_{halt}$ and $\omega_q$ as models, rank 0 of $\omega_{halt}$ in the posterior ranking function implies that $T$ has reached a halting state. Ranking functions $\kappa$ which are the result of such a revision are called **PEEK** and obey the following characteristics

\[
\kappa(\omega_q) = 0 \quad 1 \leq \kappa(\omega_q) \leq n \\
\kappa(\omega_q) = 1 \quad n + 1 < \kappa(\omega_{pos})
\]

and $\text{enc}^{-1}(\kappa(\omega_{pos}) - (n + 2)) \neq \emptyset$. For such a ranking function let $\text{conf}(\kappa) = \langle q_n(\omega_0), \text{enc}^{-1}(\kappa(\omega_{pos}) - (n + 2)) \rangle$ denote the represented configuration.

**TRANS.** A **TRANS** ranking function $\kappa$ is given if:

\[
\begin{aligned}
\kappa(\omega_0) &= 0 & 1 \leq \kappa(\omega_q) &\leq n \\
\kappa(\omega_{pos}) &> n & 2 \leq \kappa(\omega_{halt}) &\leq n
\end{aligned}
\]

For such a ranking function $\kappa$ we define $\text{post-state}(\kappa) = q_n(\omega_q)$. Moreover, let $\text{conf}(\kappa) = \langle q_n(\omega_0) - \kappa(N), \text{enc}^{-1}(\kappa(\omega_{pos}) - (n + 1)) \rangle$ denote the represented configuration (before the transition). Note that $\kappa(\omega_q)$ encodes the posterior state, but the current (prior) state is reconstructible from $\kappa(\omega_q)$ and the difference between $n$ and $\kappa(\omega_{halt})$.

**POST.** Any ranking function $\kappa$ with

\[
\begin{aligned}
\kappa(\omega_0) &= 0 & 1 \leq \kappa(\omega_q) &\leq n \\
\kappa(\omega_{halt}) &= n & n < \kappa(\omega_{pos})
\end{aligned}
\]

and with $\text{enc}^{-1}(\kappa(\omega_{pos}) - (n + 1)) \neq \emptyset$. Such a ranking function represents the configuration $\text{conf}(\kappa) = \langle q_n(\omega_q), \text{enc}^{-1}(\kappa(\omega_{pos}) - (n + 1)) \rangle$.
Simulation is performed by continuously revising by specific formulas (Algorithm 1). With \( \varphi_{q, \text{pos}} \) we denote a formula with \( \{\varphi_{q, \text{pos}}\} = \{\omega_0, \omega_{\text{pos}}\} \). This extends analogously to \( \varphi_{q, \text{halt}}, \varphi_{0, q, \text{halt}} \) and \( \varphi_{0, q, \text{pos}} \).

**Definition 8.** Let \( f : \Gamma^* \rightarrow \Gamma^* \) be a partial function computed by a Turing machine \( T \). A ranking function-based belief change operator \( \circ \) is said to simulate \( T \) if \( \text{SimulateTM}(T, \circ, \gamma) \) (Algorithm 1) yields the output of \( \gamma \) by \( T \) for all \( \gamma \in \Gamma^* \). If \( \circ \) simulates \( T \), then we say \( \circ \) computes \( f \).

### 4.4 Main Theorem

We will now provide a construction for an AGM revision operator \( \ast \) which simulates a given Turing machine and show correctness of the construction.

**Theorem 9.** Every Turing computable function \( f \) is computable by a ranking function-based AGM revision operator that satisfies the postulates (DP1)–(DP4).

**Proof.** Let \( T = \langle Q, \Gamma, \delta, q_s \rangle \) be a (deterministic) Turing machine computing \( f \). We construct a ranking function-based AGM revision operator \( \ast \) and an assignment \( \Psi \mapsto (\kappa, \beta) \). We start by letting \( \Psi \mapsto (\kappa, \beta) \) be a bijective function such that for each ranking function \( \kappa \) there exist exactly two epistemic states \( \Psi^\kappa, \Psi^\kappa_\beta \in E \) with \( \Psi^\kappa = [k] \), \( \Psi^\kappa_\beta = k \) and \( \beta_{q, k} = \top \) and with \( \Psi^\kappa_\beta = \emptyset \), \( \beta_{q, k} = k \) and \( \beta_{q, k} = \bot \). Such a mapping exists by the properties of \( E \). In the following we ease notation and just write \( \kappa \) for \( \Psi^\kappa \) and \( \kappa_\beta \) for \( \Psi^\kappa_\beta \).

Next, we provide how \( \ast \) behaves in all cases appearing in runs of \( \text{SimulateTM}(T, \ast, \gamma) \) (Algorithm 1), depending on the ranking function type of \( \kappa \):

**CONF.** We consider two different cases. These case of \( \alpha = \varphi_{q, \text{halt}} \). For this case let \( \ast \) yield the following ranking function \( \kappa \ast \alpha \):

\[
\kappa \ast \alpha(\omega_0) = 1 \\
\kappa \ast \alpha(\omega_{\text{halt}}) = \kappa(\omega_{\text{halt}}) - (i - j) \\
\kappa \ast \alpha(\omega_q) = j \\
\kappa \ast \alpha(\omega_{\text{pos}}) = \kappa(\omega_{\text{pos}}) - (i - j)
\]

Note that \( \kappa \ast \alpha \) is a \textbf{PEEK} ranking function and \( \text{conf}(\kappa) = \text{conf}(\kappa \ast \alpha) \). The case of \( \alpha = \varphi_{0, \text{pos}} \). Let \( \kappa \beta \) be a ranking function representing a configuration \( C = \langle q_j, \langle t_1, t_c, t_r \rangle \rangle \) of \( T \). Let \( \langle q_j, \langle t_1, t_c, t_r \rangle \rangle \) be the successor configuration provided by \( \delta \). Then \( \ast \) yields the following ranking function \( \kappa \ast \alpha \):

\[
\kappa \ast \alpha(\omega_0) = 0 \\
\kappa \ast \alpha(\omega_{\text{halt}}) = \kappa(\omega_{\text{halt}}) - (i - j) \\
\kappa \ast \alpha(\omega_q) = n - \kappa(\omega_{\text{halt}}) \\
\kappa \ast \alpha(\omega_{\text{pos}}) = \kappa(\omega_{\text{pos}}) - 1
\]

Note that \( \kappa \ast \alpha \) is a \textbf{TRANS} ranking function with \( \text{post-state}(\kappa \ast \alpha) = q_j \) and \( \text{conf}(\kappa) = \text{conf}(\kappa \ast \alpha) \).

**PEEK.** Based on Algorithm 1 we consider \( \alpha = \varphi_{0, \text{pos}} \). Then \( \ast \) is constructed such that it yields \( \kappa \ast \alpha \) by input \( \kappa \) and \( \alpha \) with:

\[
\kappa \ast \alpha(\omega_0) = 0 \\
\kappa \ast \alpha(\omega_{\text{halt}}) = n - \kappa(\omega_{\text{halt}}) \\
\kappa \ast \alpha(\omega_q) = \kappa(\omega_q) \\
\kappa \ast \alpha(\omega_{\text{pos}}) = \kappa(\omega_{\text{pos}}) - 1
\]

The ranking function \( \kappa \ast \alpha \) is of \textbf{CONF} and \( \text{conf}(\kappa) = \text{conf}(\kappa \ast \alpha) \).

**TRANS.** Let \( \text{post-state}(\kappa \ast \alpha) = q_j \) and \( \text{conf}(\kappa) = \langle q_j, \langle t_1, t_c, t_r \rangle \rangle \) and successor configuration \( C = \langle q_j, \langle t_1, t_c, t_r \rangle \rangle \). For \( \alpha = \varphi_{0, \text{halt}} \) let \( \ast \) yield the ranking function \( \kappa \ast \alpha \) given by:

\[
\kappa \ast \alpha(\omega_0) = 0 \\
\kappa \ast \alpha(\omega_{\text{halt}}) = \kappa(\omega_{\text{halt}}) \\
\kappa \ast \alpha(\omega_q) = \kappa(\omega_q) \\
\kappa \ast \alpha(\omega_{\text{pos}}) = n + \text{enc}(C) + 1
\]

Note that \( \kappa \ast \alpha \) is a \textbf{POST} ranking function which represents the posterior configuration \( C \), i.e., \( C = \text{conf}(\kappa \ast \alpha) \).

**POST.** As last case consider \( \alpha = \varphi_{0, \text{pos}} \). We define \( \ast \) as follows:

\[
\kappa \ast \alpha(\omega_0) = 0 \\
\kappa \ast \alpha(\omega_{\text{halt}}) = n \\
\kappa \ast \alpha(\omega_q) = \kappa(\omega_q) \\
\kappa \ast \alpha(\omega_{\text{pos}}) = \kappa(\omega_{\text{pos}})
\]

The resulting \( \kappa \ast \alpha \) is a \textbf{CONF} ranking function.

Consideration of Algorithm 1 yields that all operations \( \ast \) performed in Algorithm 1 are covered by the cases described here. In every step of Algorithm 1 every ranking function constructed is of one of the four types and only parameters treated above are applied. In particular, observe that in Lines 6 to 8 a complete transition to a new configuration is performed. Clearly, if \( \omega_{\text{halt}} \in [\kappa] \), where \( \kappa \) is of type \textbf{PEEK}, then \( T \) has also reached the halting state and \( \kappa \) represents the content of the tape of \( T \) in that configuration. Thus, Line 11 of Algorithm 1 yields the output of \( T \).

Next, we complete \( \ast \) to a ranking function-based belief change operator, by setting \( \kappa \ast \alpha = \kappa_\alpha \) for all remaining unspecified cases of \( \ast \), where \( \kappa_\alpha \) is given as follows: If \( \alpha \) is consistent, let \( \kappa_\alpha = \text{conf}(\kappa) \) denote the ranking function with:

\[
\kappa_\alpha(\omega) = \begin{cases} 
\kappa(\omega) - \min_{\omega' \in [\alpha]} (\kappa(\omega')) & \text{if } \omega \in [\alpha] \\
\kappa(\omega) + 1 & \text{if } \omega \notin [\alpha]
\end{cases}
\]

If \( \alpha \) is inconsistent, let \( \kappa_\alpha = \kappa_\bot \). A careful examination shows that \( \ast \) satisfies the quantitative postulates:

(Q1) if \( \omega, \omega' \in [\alpha] \), then \( \kappa(\omega) - \kappa(\omega') = \kappa \ast \alpha(\omega) - \kappa \ast \alpha(\omega') \)

(Q2) if \( \omega, \omega' \in [\neg \alpha] \), then \( \kappa(\omega) - \kappa(\omega') = \kappa \ast \alpha(\omega) - \kappa \ast \alpha(\omega') \)

(Q3) if \( \omega \in [\alpha] \) and \( \omega' \in [\neg \alpha] \), then \( \kappa(\omega) < \kappa(\omega') \Rightarrow \kappa \ast \alpha(\omega) < \kappa \ast \alpha(\omega') \)

(Q4) if \( \omega \in [\neg \alpha] \) and \( \omega' \in [\neg \alpha] \), then \( \kappa(\omega) \leq \kappa(\omega') \Rightarrow \kappa \ast \alpha(\omega) \leq \kappa \ast \alpha(\omega') \)

Consequently, by Proposition 7 and Proposition 4, \( \ast \) is an AGM revision operator that satisfies (DP1)–(DP4).

Theorem 9 gives rise to the central observation.

**Observation.** Iterated belief change is Turing complete.

### 5 Conclusions and Future Work

In this paper we showed that iterated belief change is Turing complete; this includes iterated AGM revision. In particular, we showed how Turing machines can be encoded in a belief revision operator over ranking functions. The results of Turing-completeness still hold when assuming the Darwiche-Pearl iteration principles for belief revision.

It is reasonable that (in certain situations) iterated belief change should be limited in its computational power, e.g., when modeling human reasoning (Ragni et al. 2020). In future work, we will identify such situations formally and investigate computational properties and restrictions by means of iteration postulates.
Acknowledgements

We thank the anonymous reviewers for their valuable hints and comments that helped us to improve the paper. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grants BE 1700/9-1 and BE 1700/10-1 awarded to Christoph Beierle as part of the priority program "Intentional Forgetting in Organizations" (SPP 1921). Kai Sauerwald is supported by the grants BE 1700/9-1 and BE 1700/10-1.

References


