Artificial Intelligence 331 (2024) 104112

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

Embedding justification theory in approximation fixpoint theory [☆,☆☆]

Simon Marynissen [a,b,*], Bart Bogaerts [b], Marc Denecker [a]

[a] *KU Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium*
[b] *Vrije Universiteit Brussel, Pleinlaan 9, B-1050 Brussels, Belgium*

A R T I C L E I N F O

Keywords:
Justification theory
Approximation fixpoint theory
Knowledge representation

A B S T R A C T

Approximation Fixpoint Theory (AFT) and Justification Theory (JT) are two frameworks to unify logical formalisms. AFT studies semantics in terms of fixpoints of lattice operators, and JT in terms of so-called justifications, which are explanations of why certain facts do or do not hold in a model. While the approaches differ, the frameworks were designed with similar goals in mind, namely to study the different semantics that arise in (mainly) non-monotonic logics. The first contribution of our current paper is to provide a formal link between the two frameworks. To be precise, we show that every justification frame induces an approximator and that this mapping from JT to AFT preserves all major semantics. The second contribution exploits this correspondence to extend JT with a novel class of semantics, namely *ultimate semantics*: we formally show that ultimate semantics can be obtained in JT by a syntactic transformation on the justification frame, essentially performing a sort of resolution on the rules.

1. Introduction

In the 1980s and 90s, the area of non-monotonic reasoning (NMR) saw fierce debates about formal semantics. In several subareas, researchers sought to formalise common-sense intuitions about knowledge of introspective agents. In these areas, appeals to similar intuitions were made, resulting in the development of similar mathematical concepts. Despite the obvious similarity, the precise relation between these concepts remained elusive. In this paper, we are concerned with two formal theories that were developed to unify semantics of (mostly non-monotonic) logics, namely Approximation Fixpoint Theory (AFT) and Justification Theory (JT).

1.1. *Approximation fixpoint theory*

Approximation Fixpoint Theory (AFT) was founded by Denecker et al. [21] as a way of unifying semantics that emerged in different subareas of non-monotonic reasoning. The main contribution of AFT was to demonstrate that, by moving to an algebraic setting, the common principles behind these concepts can be isolated and studied in a general way. This breakthrough allowed results that were achieved in the context of one of these languages to be easily transferred to another. In the early stages, AFT was applied to default logic [47], auto-epistemic logic [46], and logic programming [51,21,22]. In recent years, interest in AFT

☆ This research received partial funding from the FWO Flanders project G0B2221N.
☆☆ This research is partially funded by the Flemish AI impulse program.
* Corresponding author at: Vrije Universiteit Brussel, Pleinlaan 9, B-1050 Brussels, Belgium.
E-mail addresses: simon.marynissen@kuleuven.be (S. Marynissen), bart.bogaerts@vub.be (B. Bogaerts), marc.denecker@kuleuven.be (M. Denecker).

https://doi.org/10.1016/j.artint.2024.104112
Received 4 May 2022; Received in revised form 6 March 2024; Accepted 7 March 2024

Available online 16 March 2024

has gradually increased, with applications in various other domains, encompassing abstract argumentation [49,5], extensions to deal with inconsistencies [3], causality [13], access control [53], higher-order logic programs [16], active integrity constraints [6], stream reasoning [1], constraint languages for the semantic web [8], as well as neuro-symbolic logic programming [2].

The foundations of AFT lie in Tarski's fixpoint theory of monotone operators on a complete lattice [50]. AFT demonstrates that by moving from the original lattice L to the bilattice L^2, Tarski's theory can be generalised into a fixpoint theory for arbitrary (i.e., also non-monotone) operators. Crucially, all that is required to apply AFT to a formalism and obtain several semantics is to define an appropriate approximating operator $L^2 \to L^2$ on the bilattice; the algebraic theory of AFT then takes care of the rest. For instance, to characterise the major logic programming semantics using AFT, the approximating operator is nothing else than Fitting's well-known four-valued immediate consequence operator [29]. The (partial) stable fixpoints of that operator (as defined by AFT) are exactly the (partial) stable models of the original program; the well-founded fixpoint of the operator is the well-founded model of the program, etc. All well-known semantics of logic programming correspond to distinguished types of fixpoints of Fitting's operator defined in the AFT framework!

1.2. Justification theory

Building on an old semantic framework for (abductive) logic programming [20], Denecker et al. [19] defined an abstract theory of *justifications* suitable for describing the semantics of a range of logics in knowledge representation, computational and mathematical logic, including logic programs, argumentation frameworks and nested least and greatest fixpoint definitions. Justifications provide a refined way of describing the semantics of a logic: they not only define whether an interpretation is a model (under a suitable semantics) of a theory, but also *why*.

Justifications — albeit not always in the exact formal form as described by Denecker et al. [19] — have appeared in different ways in different areas. The stable semantics for logic programs was defined in terms of justifications [26,48]. Moreover, an algebra for combining justifications (for logic programs) was defined by Cabalar et al. [15]; and justifications are underlying provenance systems in databases [18].

Next to these theoretic benefits, justifications are also used in implementations of answer set solvers: they form the basis of the so-called source-pointer approach in the unfounded set algorithm [30], and turned out to be key in analyzing conflicts in the context of lazy grounding [14]), as well as to improve parity game solvers [37].

1.3. Correspondence

AFT and justification theory were designed with similar intentions in mind, namely to unify different (mainly non-monotonic) logics. One major difference between them is that justification theory is defined logically while AFT is defined purely algebraically. This makes justification frameworks less abstract and easier to grasp, but also in a certain sense less general, as demonstrated by the fact that logics such as autoepistemic logic have no justification semantics (yet). On the other hand, justification theory has a larger freedom to define semantics by providing a branch evaluation, as well as the notion of nesting [19], which captures the semantics of nested least and nested greatest fixpoint definitions [33] as shown by Marynissen [40].

Despite the differences, certain correspondences between the theories show up: several definitions in justification frameworks seem to have an algebraical counterpart in AFT. This is evident from the fact that many results on justifications are formulated in terms of fixpoints of the support operator (see Definition 4.5) that happens, for the case of logic programming, to coincide with (Fitting's, 2002, three-valued version of) the immediate consequence operator for logic programs. Of course, now the question naturally arises whether this correspondence can be made formal, i.e., whether it can formally be shown that semantics induced by justification theory will always coincide with their equally-named counterpart in AFT. If the answer is positive, this will allow us to translate results between the two theories. Formalising this correspondence is the key contribution of the current paper.

1.4. Contributions

Our contributions can be summarised as follows:

- In Section 3, we provide some novel results for JT. While the main purpose of these results is to support the theorems of Section 4, they also directly advance the state of JT. In this section, we give an alternative (and arguably, easier) characterization of splittable branch evaluations and use it to get new results on how justifications can be "glued together". Afterwards, we show how different semantics induced by JT relate, and we resolve a discrepancy that exists between different definitions of so-called *stable and supported branch evaluations* in prior work. We formally prove that the different circulating definitions of these branch evaluations indeed induce the same semantics.
- In Section 4, we turn our attention to the key contribution of the paper, namely embedding JT in AFT. To do this, we proceed as follows. First, we show that under minor restrictions, each justification frame (intuitively, this is a set of rules that describe when a positive or negative fact is true), can be transformed into an approximator. Next, we show that for each of the most common *branch evaluations* (these are mathematical structures that are used to associate semantics to a justification frame), the semantics induced by JT is the same as the equally-named semantics on the AFT side. Establishing this result is of particular importance for the future development of JT, since this result immediately makes a large body of theoretical results developed in the context of AFT readily available for JT, as well as all its future application domains, including results on stratification [55,7], predicate

introduction [56], and knowledge compilation [9]. On the other hand, from the context of AFT, the embedding of JT can serve as inspiration for developing more general algebraic explanation mechanisms.

- To illustrate how this connection can be exploited for further extending the theory of justifications by transferring results from AFT to JT, we turn our attention to *ultimate semantics*. In the context of AFT, Denecker and his coauthors have realised that a single operator can have multiple approximators and that the choice of approximator influences the induced semantics. They also showed that — when staying in the realm of consistent AFT — every operator induces a most precise approximator, and called this *the ultimate approximator* [23]. In Section 5, we transfer this idea to JT. We show there that by means of a simple transformation[1] on the justification frame, we can obtain ultimate semantics. Importantly, since this transformation is defined independently of the branch evaluation at hand, ultimate semantics are not just induced for the semantics that have a counterpart in AFT, but for all conceivable current and future branch evaluations as well.

While establishing the connection between JT and AFT has no direct implications on fields such as logic programming and abstract argumentation, for which characterizations in terms of justifications as well as in terms of approximating operators already existed, there are several benefits to establishing this connection. On the one hand, we obtain a deeper understanding of the two frameworks (JT and AFT), and as already mentioned above, it allows transferring results between fields, which we illustrate by means of the ultimate semantics. On the other hand, it also means that whenever new application domains of JT arise, an operator-based characterization of the semantics in those fields is obtained for free. This would mean for instance that for such application domains, we obtain constructive characterizations of the semantics at hand (e.g., well-founded inductions [24]), without additional effort.

Publication history The short version of this paper was presented at IJCAI 2021 [43].[2] This paper extends the short version with proofs, extra examples, and an expanded exposition. Most of these extensions are also included in the first author's PhD thesis [40].

2. Preliminaries: justification theory

In this section, we recall the formalisation of Justification Theory of Marynissen et al. [42].

Truth values are denoted $\mathbf{t}$ (true), $\mathbf{f}$ (false) and $\mathbf{u}$ (unknown); we write $\mathcal{L}$ for $\{\mathbf{t},\mathbf{f},\mathbf{u}\}$. We make use of two orders on $\mathcal{L}$, the *truth order* $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$ and the *precision order* $\mathbf{u} \leq_p \mathbf{f}, \mathbf{t}$. JT starts with a set $\mathcal{F}$, referred to as a *fact space*, such that $\mathcal{L} \subseteq \mathcal{F}$; the elements of $\mathcal{F}$ are called *facts*. We assume that $\mathcal{F}$ is equipped with an involution $\sim : \mathcal{F} \to \mathcal{F}$ (i.e., a bijection that is its own inverse) such that $\sim\mathbf{t} = \mathbf{f}$, $\sim\mathbf{u} = \mathbf{u}$, and $\sim x \neq x$ for all $x \neq \mathbf{u}$. Moreover, we assume that $\mathcal{F} \setminus \mathcal{L}$ is partitioned into two disjoint sets $\mathcal{F}_+$ and $\mathcal{F}_-$ such that $x \in \mathcal{F}_+$ if and only if $\sim x \in \mathcal{F}_-$ for all $x \in \mathcal{F} \setminus \mathcal{L}$. Elements of $\mathcal{F}_+$ are called *positive* and elements of $\mathcal{F}_-$ are called *negative* facts. An example of a fact space is the set of literals over a propositional vocabulary Σ extended with $\mathcal{L}$ where $\sim$ maps a literal to its negation. For any set A we define $\sim A$ to be the set of elements of the form $\sim a$ for $a \in A$. We distinguish two types of facts: *defined* and *open* facts. The former are accompanied by a set of rules that determine their truth value. The truth value of the latter is not governed by the rule system but comes from an external source or is fixed (as is the case for logical facts).

Definition 2.1. A *justification frame* $\mathcal{JF}$ is a tuple $\langle \mathcal{F}, \mathcal{F}_d, R \rangle$ such that

- $\mathcal{F}_d$ is a subset of $\mathcal{F}$ closed under $\sim$, i.e., $\sim\mathcal{F}_d = \mathcal{F}_d$; facts in $\mathcal{F}_d$ are called *defined*;
- no logical fact is defined: $\mathcal{L} \cap \mathcal{F}_d = \emptyset$;
- $R \subseteq \mathcal{F}_d \times 2^{\mathcal{F}} \setminus \{\emptyset\}$;
- for each $x \in \mathcal{F}_d$ there is at least one element $(x, A) \in R$.

A fact $x \in \mathcal{F} \setminus \mathcal{F}_d$ is called *open* (or sometimes a parameter). The set of open facts is denoted $\mathcal{F}_o$. Notice that $\mathcal{L} \subseteq \mathcal{F}_o$, i.e., truth values are also called open facts. An element $(x, A) \in R$ is called a *rule* with *head* x and *body* (or *case*) A. The set of cases of x in $\mathcal{JF}$ is denoted $\mathcal{JF}(x)$. Rules $(x, A) \in R$ are denoted as $x \leftarrow A$ and if $A = \{y_1, \dots, y_n\}$, we often write $x \leftarrow y_1, \dots, y_n$.

Logic programs as justification frames Justification theory was developed partly for the semantic study of logic programming. The relationship between logic programs and justification frames is the following. The negation operator $\sim$ corresponds to negation *as failure* `not` in LP, and positive and negative facts $x \in \mathcal{F}_d$ correspond to LP literals x and `not` x. Justification frames generalize propositional LP in three ways: (1) a rule body in a justification frame may be an arbitrary set rather than a finite one (as seems suitable for an abstract semantic framework); (2) contrary to logic programs, justification frames may possess open facts; (3) rules in justification frames may have $\sim$ in the head.

The idea of open facts that are not in $\mathcal{L}$ is that their truth value is not constrained by the justification framework. In contrast, every atom x in a logic program is constrained, even if there is no rule with x as head[3]: in this case, x is false according to the logic program. In justification theory, this is to be modelled by including the rule $x \leftarrow \mathbf{f}$ in R.

[1] Essentially, this transformation performs some sort of case splitting.

[2] Where it received a *Distinguished Paper Award*.

[3] There are also logic programming formalisms that have parameters; for instance the *input atoms* in the terminology of Janhunen et al. [35] correspond to parameters in justification theory.

Rules in justification frames may contain $\sim$ in the head of rules, whereas standard LP rules do not contain `not` in the head.[4] However, in concrete instances of JT, the rules $\sim x \leftarrow$ are strictly derived from those for the positive facts x by a sort of completion process. The derivation is described by a technique called *complementation* originally proposed by Denecker et al. [19]. This is a generic mechanism turning a set of rules for x into a set of rules for $\sim x$ with the aim to provide explanation for why x is *not* justified. To define complementation, we first define *selection functions* for x. A selection function for x is a mapping $s : \mathcal{JF}(x) \to \mathcal{F}$ such that $s(A) \in A$ for all rules of the form $x \leftarrow A$. Intuitively, a selection function chooses an element from the body of each rule for x. For a selection function s, the set $\{s(A) \mid A \in \mathcal{JF}(x)\}$ is denoted by $\mathrm{Im}(s)$.[5]

Definition 2.2. For a set of rules R, we define R^* to be the set of rules of the form $\sim x \leftarrow \sim \mathrm{Im}(s)$ for $x \in \mathcal{F}_d$ that has rules in R and s a selection function for x. The *complementation* of $\mathcal{JF}$ is defined as $\langle \mathcal{F}, \mathcal{F}_d, R \cup R^* \rangle$.

Since the complementation contains a rule for $\sim x$ for every selection function for x, the complementation intuitively contains all possible ways of "blocking" the rules for x.[6]

Example 2.3. If $R = \{x \leftarrow a, b;\ \ x \leftarrow c;\}$, then $R^* = \{\sim x \leftarrow \sim a, \sim c;\ \ \sim x \leftarrow \sim b, \sim c\}$. ▲

In summary, a (propositional) logic program π corresponds to a justification frame $\mathcal{JF}_\pi = \langle \mathcal{F}, \mathcal{F}_d, R \rangle$ such that $\mathcal{F}_d$ is the set of positive and negative literals of π and R is the set of rules of π extended with, first, rules $p \leftarrow \mathbf{f}$ for undefined program atoms p of π and, second, the complementation of this rule set. Formally, $R = \pi_c \cup \pi_c^*$ where π_c is $\pi \cup \{p \leftarrow \mathbf{f} \mid p$ is an undefined program atom in $\pi\}$. Further down, we will see how different semantics of LP arise by the choice of different *branch evaluations*.

Justifications The main semantic objects of justification theory are so-called *justifications*.

Definition 2.4. A *directed graph* is a pair (N, E) where N is a set of nodes and $E \subseteq N \times N$ is the set of edges. An *internal* node is a node with outgoing edges. A *leaf* is a non-internal node.

Definition 2.5. Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \rangle$ be a justification frame. A *justification* J in $\mathcal{JF}$ is a directed graph (N, E) such that every node $n \in N$ occurs on an edge in E and for every internal node $n \in N$, the rule $n \leftarrow \{m \mid (n, m) \in E\}$ is an element of R. We say that J *justifies* n if n is an internal node of J.

Remark 2.6. In some papers, a distinction is made between tree-like and graph-like justifications [42]. In a tree-like justification, a fact may have many occurrences while in a graph-like justification, it has at most one. Since we restrict attention to the latter, we shall just use the term *justification* to mean graph-like justifications. Our main result in this paper is an embedding of justification theory into AFT that preserves several semantics. For each of these semantics, graph-like and tree-like justifications have been shown to be equivalent [40], hence all our results are valid for tree-like justifications as well. ▲

A justification J can be viewed as a partial function $\mathcal{F}_d \to R$ such that $J(n)$ is the rule $n \leftarrow \{m \mid (n, m) \in E\}$. Its domain $\mathrm{dom}(J)$ is the set of justified facts.

A justification is *complete* if $\mathrm{dom}(J) = \mathcal{F}_d$. A justification is *locally complete* if it has no leaves in $\mathcal{F}_d$. We write $\mathfrak{J}_{\mathcal{JF}}(x)$ to denote the set of locally complete justifications that have an internal node x (and write $\mathfrak{J}(x)$ if $\mathcal{JF}$ is clear from the context.

A justification J with internal fact x is called *proper* for x if all facts of J are reachable from x in the graph J. It is straightforward to show that by reducing a non-proper justification J for x to the set of nodes reachable from x, a proper justification of x is obtained, denoted $J|_x$. If J is locally complete, then so is $J|_x$. We write $\mathfrak{PJ}(x)$ to denote the set of proper justifications of x.

Example 2.7. Take $\mathcal{F}_d = \{x, \sim x, y, \sim y\}$, $\mathcal{F}_o = \{a, \sim a, b, \sim b\} \cup \mathcal{L}$, and R the complementation of

$$\left\{ \begin{array}{l} x \leftarrow y, a \\ y \leftarrow y, b \end{array} \right\}.$$

Then

[4] Answer Set Programming supports a second negation operator – called *explicit negation* which can appear in head of rules. However, $\sim$ is no match with $-$, as can be seen in the rest of this paragraph. Semantic extensions of ASP also extend in which default negation can appear in the head of a rule [34].

[5] The existence of selection functions for arbitrary justification frames depends on the axiom of choice.

[6] Complementation as described here is similar in flavour to Clark's completion [17], except that the result is not a propositional theory, but a set of rules for positive and negative facts. As we shall see later, this is useful for defining a variety of semantics, formalizing different sorts of reasons *why* certain facts are true or false.

$$\begin{tikzcd}
& x \arrow[dl] \arrow[dr] & \\
y \arrow[loop left] \arrow[d] & & a \\
b & &
\end{tikzcd}$$

is a locally complete justification in $\langle \mathcal{F}, \mathcal{F}_d, R \rangle$ since a and b are open facts. It is proper only for x. ▲

Let $\mathcal{JF}$ be a justification frame.

Definition 2.8. A *$\mathcal{JF}$-path* is a finite or infinite sequence $x_0 \to x_1 \to \cdots$ of non-zero length, consisting of defined facts potentially ended with an open fact. A $\mathcal{JF}$-path is called a *$\mathcal{JF}$-branch* if it cannot be extended at its tail, that is, if it is infinite or ending with an open fact. If $\mathbf{b}$ is the branch $x_0 \to x_1 \to \cdots$, we write $\sim\mathbf{b}$ for $\sim x_0 \to \sim x_1 \to \cdots$. A *tail* of a branch $\mathbf{b}$ is a branch $x_i \to x_{i+1} \to \cdots$ for some $i \geq 0$.

Definition 2.9. For a $\mathcal{JF}$-justification J, a *J-branch* $\mathbf{p}$ starting from an internal node $x \in \mathcal{F}_d$ is a path through J starting from x that is infinite or ends in a leaf of J. We write $B_J(x)$ to denote the set of J-branches starting from x.

$\mathcal{JF}$-branches $\mathbf{b}$ are $\mathcal{JF}$-paths that cannot be extended any further. Similarly, J-branches $\mathbf{p}$ are paths through J that cannot be extended any further in J. Notice that a J-branch that ends in a defined leaf of J is a $\mathcal{JF}$-path, not a $\mathcal{JF}$-branch. However, all J-branches of locally complete justifications J are $\mathcal{JF}$-branches. Each J-branch starting at internal node x of J is clearly also a $J|_x$-branch; hence, $B_J(x) = B_{J|_x}(x)$.

Definition 2.10. We say that an infinite branch is *positive (respectively negative)* if it consists of positive facts (respectively negative facts). An infinite branch is called a *eventually positive (respectively negative)* if it has a positive (respectively negative) tail. If it is neither eventually positive nor eventually negative, it is called *mixed*.

Notice that a mixed branch necessarily contains infinitely many positive and infinitely many negative facts.
Intuitively, a justification formally captures a potential answer to the question "Why?".

Why is a fact x_0 true? Because the facts in the body of a rule $x_0 \leftarrow A_0$ are true. But, why is some fact $x_1 \in A_0$ true? Because the facts in the body of a rule $x_1 \leftarrow A_1$ are true. But why is some fact $x_2 \in A_1$ true? Etc.

A (locally complete) justification collects all answers of this exhaustive, iterated process of posing "why?" questions.[7] A branch $x_0 \to x_1 \to x_2 \to \ldots$ in a justification formalizes a linear sequence "x_0 because x_1 because x_2 because ..." which goes on until the question can be decided. But justifications do not capture how such a process is to be evaluated. E.g., when can we stop? How to evaluate infinite branches? There are many mathematical ways to do this. They are captured in different kinds of *branch evaluations*.

Branch evaluations The semantics for justification frames is defined by a branch evaluation.

Definition 2.11. A *branch evaluation* $\mathcal{B}$ is a mapping that maps any $\mathcal{JF}$-branch to an element of $\mathcal{F}$ for all justification frames $\mathcal{JF}$. A branch evaluation $\mathcal{B}$ *respects negation* if $\mathcal{B}(\sim\mathbf{b}) = \sim\mathcal{B}(\mathbf{b})$ for any branch $\mathbf{b}$. A justification frame $\mathcal{JF}$ together with a branch evaluation $\mathcal{B}$ forms a *justification system* $\mathcal{JS}$, which is presented as a quadruple $\langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$.

Some branch evaluations of interest are given below. All of them respect negation.

Definition 2.12 *($\mathcal{B}_{\text{sp}}$, $\mathcal{B}_{\text{KK}}$, $\mathcal{B}_{\text{st}}$, $\mathcal{B}_{\text{wf}}$).* The *supported* branch evaluation $\mathcal{B}_{\text{sp}}$ maps $x_0 \to x_1 \to \cdots$ to x_1. The *Kripke-Kleene* branch evaluation $\mathcal{B}_{\text{KK}}$ maps finite branches to their last element and infinite branches to $\mathbf{u}$. The *well-founded* branch evaluation $\mathcal{B}_{\text{wf}}$ maps finite branches to their last element and infinite branches to $\mathbf{t}$ if they have a negative tail, to $\mathbf{f}$ if they have a positive tail and to $\mathbf{u}$ otherwise. The *stable* branch evaluation $\mathcal{B}_{\text{st}}$ maps a branch $x_0 \to x_1 \to \cdots$ to the first element that has a different sign than x_0 if it exists; otherwise $\mathbf{b}$ is mapped to $\mathcal{B}_{\text{wf}}(\mathbf{b})$. I.e., finite branches without sign switch are mapped to their last element; infinite positive branches are mapped to $\mathbf{f}$ and infinite negative branches to $\mathbf{t}$.

Example 2.13. Consider the following branches:

$\mathbf{b}_1 : x \to y \to a,$

[7] This process of iterated "Why?" questions will sound familiar to many parents, especially how exhaustive it can be!

$$\mathbf{b}_2 : x \to y \to {\sim}z \to x \to y \to {\sim}z \to \dots,$$

$$\mathbf{b}_3 : x \to y \to z \to x \to y \to z \to \dots.$$

In that case, it holds that $\mathcal{B}_{\text{sp}}(\mathbf{b}_1) = \mathcal{B}_{\text{sp}}(\mathbf{b}_2) = \mathcal{B}_{\text{sp}}(\mathbf{b}_3) = y$, since $\mathcal{B}_{\text{sp}}$ maps all branches to their second element. We see that $\mathcal{B}_{\text{KK}}(\mathbf{b}_1)$ is a, since $\mathcal{B}_{\text{KK}}$ maps finite branches to their last element, and $\mathcal{B}_{\text{KK}}(\mathbf{b}_2) = \mathcal{B}_{\text{KK}}(\mathbf{b}_3) = \mathbf{u}$. For the well-founded branch evaluation, we see that $\mathcal{B}_{\text{wf}}(\mathbf{b}_1) = a, \mathcal{B}_{\text{wf}}(\mathbf{b}_2) = \mathbf{u}$, and $\mathcal{B}_{\text{wf}}(\mathbf{b}_3) = \mathbf{f}$. The stable branch evaluation agrees with $\mathcal{B}_{\text{wf}}$ on $\mathbf{b}_1$ and $\mathbf{b}_3$, but not on $\mathbf{b}_2$, which contains a sign switch: it holds that $\mathcal{B}_{\text{st}}(\mathbf{b}_2) = {\sim}z$. ▲

Definition 2.14. A *(three-valued) interpretation* of $\mathcal{F}$ is a function $\mathcal{I} : \mathcal{F} \to \mathcal{L}$ such that $\mathcal{I}({\sim}x) = {\sim}\mathcal{I}(x)$ for all $x \in \mathcal{F}$ and $\mathcal{I}(\ell) = \ell$ for all $\ell \in \mathcal{L}$. $\mathcal{I}(x)$ is called the value of x in $\mathcal{I}$.

Definition 2.15. Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a justification system, $\mathcal{I}$ an interpretation of $\mathcal{F}$, and J a locally complete justification in $\mathcal{JS}$. Let $x \in \mathcal{F}_d$ be a node in J. The *supported value* of $x \in \mathcal{F}_d$ by J under $\mathcal{I}$ is defined as

$$\sup_{\mathcal{B}}(x, J, \mathcal{I}) = \min_{\mathbf{b} \in B_J(x)} \mathcal{I}(\mathcal{B}(\mathbf{b})),$$

where min is with respect to $\leq_t$.

Definition 2.16. The *supported value* of $x \in \mathcal{F}$ in $\mathcal{JS}$ under $\mathcal{I}$ is defined as

$$\text{SV}_{\mathcal{JS}}(x, \mathcal{I}) = \max_{J \in \mathfrak{J}(x)} \sup_{\mathcal{B}}(x, J, \mathcal{I})$$

for $x \in \mathcal{F}_d$ and $\text{SV}_{\mathcal{JS}}(x, \mathcal{I}) = \mathcal{I}(x)$ for $x \in \mathcal{F}_o$. If $\mathcal{JS}$ consists of $\mathcal{JF}$ and $\mathcal{B}$, and $\mathcal{JF}$ is clear from context, we write $\text{SV}_{\mathcal{B}}$ for $\text{SV}_{\mathcal{JS}}$.

Since the branches starting at x in the justifications J and $J|_x$ are the same, it follows that for all justifications $J \in \mathfrak{J}(x)$, $\sup_{\mathcal{B}}(x, J, \mathcal{I}) = \sup_{\mathcal{B}}(x, J|_x, \mathcal{I})$. As a consequence, for determining the supported value of x, it suffices to consider only the proper justifications of x, that is, the elements of $\mathfrak{PJ}(x)$.

Models under justification semantics are determined by the supported value under $\mathcal{B}$.

Definition 2.17. Let $\mathcal{JF}$ be a justification frame and $\mathcal{B}$ a branch evaluation. An $\mathcal{F}$-interpretation $\mathcal{I}$ is a *$\mathcal{B}$-model of $\mathcal{JF}$* if for all $x \in \mathcal{F}$, $\text{SV}_{\langle \mathcal{JF}, \mathcal{B} \rangle}(x, \mathcal{I}) = \mathcal{I}(x)$.

A $\mathcal{B}_{\text{sp}}$, $\mathcal{B}_{\text{KK}}$, $\mathcal{B}_{\text{st}}$, or $\mathcal{B}_{\text{wf}}$-model is called a supported, Kripke-Kleene, stable, or well-founded model.

In a $\mathcal{B}$-model $\mathcal{I}$ of $\mathcal{JF}$, there is a strong consistency property between the supported values of x and ${\sim}x$:

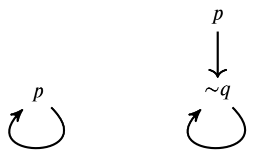
$$\text{SV}_{\mathcal{B}}({\sim}x, \mathcal{I}) = \mathcal{I}({\sim}x) = {\sim}\mathcal{I}(x) = {\sim}\,\text{SV}_{\mathcal{B}}(x, \mathcal{I})$$

As such, the existence of models imposes a strong condition on justification systems. E.g., a branch evaluation $\mathcal{B}$ mapping all branches to $\mathbf{t}$ cannot have models. E.g., a frame system with $R = \{a \leftarrow \mathbf{t}, {\sim}a \leftarrow \mathbf{t}\}$ cannot have models under any sensible branch evaluation. In Section 3.3, this consistency is studied in detail and it is shown that *complementation* is important. In the examples that follow, the rules for negative facts are derivable by complementation, and this is no coincidence.

Example 2.18. Let $\mathcal{F} = \{p, {\sim}p, q, {\sim}q\} \cup \mathcal{L}$ and take R to be the following set of rules:

$$\left\{ \begin{array}{ll} p \leftarrow p & {\sim}p \leftarrow {\sim}p, q \\ p \leftarrow {\sim}q & \\ q \leftarrow q & {\sim}q \leftarrow {\sim}q \end{array} \right\}.$$

Note that the rules for ${\sim}p$ and ${\sim}q$ can be obtained through complementation starting from the rules for p and q. The only proper locally complete justifications of p are the following:



Under $\mathcal{B}_{\text{wf}}$, the left justification has a value $\mathbf{f}$ for p, while the right justification has a value $\mathbf{t}$ for p. This means that $\text{SV}_{\mathcal{B}_{\text{wf}}}(p, \mathcal{I}) = \mathbf{t}$ for all interpretations $\mathcal{I}$ of $\mathcal{F}$. The right justification also guarantees that $\text{SV}_{\mathcal{B}_{\text{wf}}}({\sim}q, \mathcal{I}) = \mathbf{t}$ for all interpretations $\mathcal{I}$, hence the unique $\mathcal{B}_{\text{wf}}$-model is the interpretation mapping p to $\mathbf{t}$ and q to $\mathbf{f}$. ▲

Example 2.19. Justification theory does not require finiteness of the set of facts, the set of rules, nor of the bodies of rules. To illustrate this, consider the infinite set of rules

$$\left\{ \begin{array}{l} p_n \leftarrow p_{n+1} \\ p_n \leftarrow \sim p_{n+1} \\ \sim p_n \leftarrow \sim p_{n+1}, p_{n+1} \end{array} \middle| \, n \in \mathbb{N} \right\}.$$

This corresponds to (the complementation of) the logic program used by Fages [27] to prove his Theorem 3.6. The corresponding logic program $\{\dots p_n \leftarrow p_{n+1}, p_n \leftarrow \sim p_{n+1}, \dots\}$ is known to have no (exact) stable models. We now show how this can be seen using justifications. To this end, assume that $\mathcal{I}$ is a $\mathcal{B}_{\text{st}}$-model. We will show that it must be so that $\mathcal{I}(p_k) = \mathbf{u}$ for all k (hence showing there are no two-valued $\mathcal{B}_{\text{st}}$-models). We claim that

1. If $\mathcal{I}(p_k) = \mathbf{f}$, then $\mathcal{I}(p_{k'}) = \mathbf{t}$ for all $k' > k$.
2. If $\mathcal{I}(p_k) = \mathbf{t}$, then $\mathcal{I}(p_{k'}) = \mathbf{f}$ for some $k' > k$.

In other words, false atoms p_k are followed only by true atoms $p_{k+1}, \dots$, but any true atom is followed later by a false atom $p_{k'}$. This is clearly a contradiction. Thus, these two claims together entail that $\mathcal{I}(p_k)$ must be $\mathbf{u}$ for all k. We now prove the claims.

For the first claim (1), assume $\mathcal{I}(p_k) = \mathbf{f}$. Since $\mathcal{I}$ is a $\mathcal{B}_{\text{st}}$-model, $\sim p_k$ must be supported by some justification. Each (proper) justification of $\sim p_k$ has the structure

$$\begin{tikzcd}
& \sim p_k \arrow[dl] \arrow[dr] & & \\
p_{k+1} & & \sim p_{k+1} \arrow[dl] \arrow[dr] & \\
\vdots & p_{k+2} & & \sim p_{k+2} \\
& \vdots & & \vdots
\end{tikzcd}$$

Since the value of a mixed branch is the element at its first sign switch, for any justification of the above structure to evaluate to true, it must be so that $\mathcal{I}(p_{k'}) = \mathbf{t}$ for all $k' > k$.

For the second claim (2), assume $\mathcal{I}(p_k) = \mathbf{t}$. In the $\mathcal{B}_{\text{st}}$-model $\mathcal{I}$, there must be a justification that supports p_k. Each proper justification of p_k has one of these two forms:

$$\begin{tikzcd}
p_k \arrow[d] & & & p_k \arrow[d] & \\
p_{k+1} \arrow[d] & & & p_{k+1} \arrow[d] & \\
p_{k+2} \arrow[d] & & & \vdots \arrow[d] & \\
\vdots & & & p_{k'} \arrow[d] & \\
& & & \sim p_{k'+1} \arrow[dl] \arrow[dr] & \\
& & \vdots & & \vdots
\end{tikzcd}$$

The first justification consists of a single branch with a positive tail. Hence, it evaluates to false and does not support p_k. For the second justification to be true, it must be so that $\mathcal{I}(\sim p_{k'+1}) = \mathbf{t}$, i.e., that $\mathcal{I}(p_{k'+1}) = \mathbf{f}$. Hence, we indeed find that for some $k' \geq k$ that $\mathcal{I}(p_{k'+1}) = \mathbf{f}$, which confirms the second claim. ▲

2.1. Classes of branch evaluations

In this paper, we will prove properties of several classes of branch evaluations, such as *splittable* or *tail-determined* branch evaluations (as introduced by Marynissen et al. [45]); their definitions are included next.

The *length* $\ell(\mathbf{b})$ of a branch $\mathbf{b}$ is defined in the following natural way: for a finite branch $\mathbf{b} : x_0 \to x_1 \to \cdots \to x_n$, $\ell(\mathbf{b}) = n$. If $\mathbf{b}$ is infinite, we say $\ell(\mathbf{b}) = \infty$. Two branches $x_0 \to x_1 \to \cdots$ and $y_0 \to y_1 \to \cdots$ are *identical up to n* if for all $0 \leq i \leq n$, we have $x_i = y_i$.

We say a $\mathcal{JF}$-branch $\mathbf{b}: x_0 \to x_1 \to \cdots$ is *head-determined*[8] under $\mathcal{B}$ at some integer n with $0 < n < \ell(\mathbf{b}) + 1$ if for every $\mathcal{JF}$-branch $\mathbf{b}'$ identical to $\mathbf{b}$ up to n, we have $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{b}')$. This means that to determine the value of $\mathbf{b}$ under $\mathcal{B}$, we only need the first $n+1$ elements, so all relevant information is located at the beginning of the branch. On the other hand, we define $\mathbf{b}$ to be *tail-determined* under $\mathcal{B}$ at $0 \leq n < \ell(\mathbf{b})$ if

$$\mathcal{B}(\mathbf{b}) = \mathcal{B}(x_n \to x_{n+1} \to \cdots).$$

Intuitively, a branch is tail-determined if all information needed to evaluate it is located in the tail starting with the n^{th} element. A branch $\mathbf{b}$ is called *splittable* under $\mathcal{B}$ at $0 \leq n < \ell(\mathbf{b}) + 1$ if it is either head-determined or tail-determined under $\mathcal{B}$ at n. Intuitively, if a branch is splittable at n, then the information to evaluate the branch is either in its tail or in its head, but not in a combination of both. If $\mathcal{B}$ is clear from context, 'under $\mathcal{B}$' is left out.

A branch $\mathbf{b}$ is called *first head-determined* at n if n is the smallest position on $\mathbf{b}$ such that $\mathbf{b}$ is head-determined at n.

We say that a branch $\mathbf{b}$ is *totally head-determined* if it is head-determined at i for every $0 < i < \ell(\mathbf{b}) + 1$. Similarly, $\mathbf{b}$ is *totally tail-determined* if it is tail-determined at i for every $0 \leq i < \ell(\mathbf{b})$. A branch $\mathbf{b}$ is *totally splittable* if it is splittable at i for every $0 \leq i < \ell(\mathbf{b}) + 1$.

Definition 2.20. A branch evaluation $\mathcal{B}$ is called

- *splittable* if every $\mathcal{JF}$-branch is totally splittable;
- *tail-determined* if every $\mathcal{JF}$-branch is totally tail-determined;
- *head-determined* if every $\mathcal{JF}$-branch is totally head-determined;

for every justification frame $\mathcal{JF}$.

Clearly, all tail-determined or head-determined branch evaluations are splittable.

Proposition 2.21. *$\mathcal{B}_{\text{sp}}$, $\mathcal{B}_{\text{KK}}$, $\mathcal{B}_{\text{st}}$, and $\mathcal{B}_{\text{wf}}$ are splittable.*

Proof. $\mathcal{B}_{\text{sp}}$ is totally head-determined on all branches, and thus is splittable.

$\mathcal{B}_{\text{KK}}$ is splittable because it is tail-determined.

$\mathcal{B}_{\text{st}}$ is totally tail-determined on positive branches and negative branches. Take a branch

$$\mathbf{b}: x_0 \to x_1 \to \cdots$$

with a first sign switch at i. It is straightforward that $\mathbf{b}$ is head-determined at $j \geq i$ and tail-determined at $j < i$. This proves that $\mathbf{b}$ is totally splittable; hence $\mathcal{B}_{\text{st}}$ is splittable.

$\mathcal{B}_{\text{wf}}$ is splittable because it is tail-determined. □

3. Fundamental results in justification theory

In this section, we prove some results about JT that will be needed for developing our theory later on. These results resolve several issues that were left open in prior work, but turn out to be crucial for studying the relationship with AFT.

3.1. Analysis of splittable branch evaluations

This section offers novel results to analyze and simplify the concepts of tail-determined and head-determined branches, arriving at an alternative, simplified and more insightful definition of splittable branch evaluations.

To start this analysis, we rephrase the concept "head-determined" in terms of paths rather than positions of branches. We observe that a branch $\mathbf{b}$ is head-determined at position i if and only if all branches with initial segment $x_0 \to \ldots \to x_i$ have the same value as $\mathbf{b}$ under $\mathcal{B}$. This motivates the following definition:

Definition 3.1. A $\mathcal{JF}$-path $\mathbf{p}$ is *decided* under $\mathcal{B}$ if all $\mathcal{JF}$-branches with initial segment $\mathbf{p}$ have the same value under $\mathcal{B}$. We extend the domain of $\mathcal{B}$ to the set of all decided $\mathcal{JF}$-paths $\mathbf{p}$ by defining $\mathcal{B}(\mathbf{p}) = \mathcal{B}(\mathbf{b})$ for some branch $\mathbf{b}$ extending $\mathbf{p}$ (the choice of $\mathbf{b}$ does not matter).

In the sequel we will drop the prefix $\mathcal{JF}$ and talk about *paths* and *branches* whenever $\mathcal{JF}$ is clear from context. We say that a path $\mathbf{p}'$ extends a path $\mathbf{p}$ if $\mathbf{p}$ is an initial segment of $\mathbf{p}'$. According to this definition, a path extends itself.

[8] In earlier work, a head-determined branch was called *decided*, and a tail-determined branch was called *transitive*. We chose here for a slightly different naming to better reflect the properties at hand.

Example 3.2. Under the stable branch evaluation, the path

$$x \rightarrow y \rightarrow \sim z \rightarrow u$$

is decided (that is, assuming $x, y, z, u \in \mathcal{F}_+$): for any branch **b** that extends this path, $\mathcal{B}_{\text{st}}(\mathbf{b}) = \sim z$. ▲

For a finite path $\mathbf{p} = x_0 \rightarrow \ldots \rightarrow x_n$ and a path $\mathbf{p}' = x_n \rightarrow \ldots$, we define the concatenation $\mathbf{p} \circ \mathbf{p}'$ as the path with initial segment **p** and tail $\mathbf{p}'$ starting at position n. Notice that concatenation of paths and branches always produces branches.

Clearly, a branch **b** is head-determined at $i > 0$ if and only if its initial segment $x_0 \rightarrow \ldots \rightarrow x_i$ is decided. Also, every branch, finite or infinite, is an initial segment of itself and is its only extension. Therefore, every branch is decided.

A next observation is that if branch **b** is head-determined at position i then also at every position $j > i$ of **b**. Stated in terms of paths, if path **p** is decided, then each path $\mathbf{p}'$ that extends **p** is decided. Indeed, every branch that extends $\mathbf{p}'$ also extends **p** and has the same value as every branch extending **p**.

Proposition 3.3. *If a path* **p** *is decided under $\mathcal{B}$, then every extension* $\mathbf{p}'$ *of* **p** *is decided and* $\mathcal{B}(\mathbf{p}') = \mathcal{B}(\mathbf{p})$.

An interesting class of decided paths are the least ones: those that do not strictly extend other decided paths. We call them *core paths*. Each decided path **p** extends a unique core path. Mathematically speaking, this follows from the fact that the set of decided paths extended by **p** is a well-order under the "is extended by" relation and every well-order has a least element.[9] Proposition 3.3 guarantees that all paths and branches that extend a core path **p** are decided and have value $\mathcal{B}(\mathbf{p})$.

Definition 3.4. A path **p** is a *core path* under $\mathcal{B}$ if it is decided under $\mathcal{B}$ and it has no strictly smaller initial segment that is decided under $\mathcal{B}$. The core path of a path **p** is the least initial segment of **p** that is core, if such segment exists. It is denoted $core_{\mathcal{B}}(\mathbf{p})$.

Example 3.5. Assume $p, q, r \in \mathcal{F}_+$ and consider the (infinite) branch

$$r \rightarrow p \rightarrow \sim q \rightarrow p \rightarrow \sim q \rightarrow p \rightarrow \sim q \rightarrow \ldots.$$

Under $\mathcal{B}_{\text{st}}$, its core is the segment $r \rightarrow p \rightarrow \sim q$ since any branch that starts with $r \rightarrow p \rightarrow \sim q$ will be mapped to $\sim q$. Under $\mathcal{B}_{\text{sp}}$, its core is the segment $r \rightarrow p$ since any branch that starts with $r \rightarrow p$ will be mapped to p. Under $\mathcal{B}_{\text{wf}}$ and $\mathcal{B}_{\text{KK}}$, this branch is its own core path. ▲

The following proposition collects the properties of core paths observed above.

Proposition 3.6. *A path* **p** *has a core* $core_{\mathcal{B}}(\mathbf{p})$ *if and only if* **p** *is decided. All paths and branches extending a core path* **p** *are decided, have the same value as* **p**, *and the least decided path that they extend is* **p**.

Proof. Straightforward. □

These properties are relevant for totally splittable branches **b**, those branches that are head-determined or tail-determined at each position, as shown in the following theorem.

Theorem 3.7. *Let $\mathcal{B}$ be an arbitrary branch evaluation. If a totally splittable branch* $\mathbf{b} = x_0 \rightarrow \ldots$ *is head-determined at some position, then there exists a $j > 0$ such that* **b** *is tail-determined at all positions $k < j$ and head-determined at all positions $k \geq j$. In this case,* $x_0 \rightarrow \ldots \rightarrow x_j$ *is the core of* **b**. *Alternatively, if* **b** *has no head-determined positions, $\mathcal{B}$ is tail-determined at every position.*

Proof. Straightforward. □

Based on this theorem, we obtain a novel characterisation of splittable branch evaluations.

Theorem 3.8. *$\mathcal{B}$ is splittable if and only if for every core path* **p**, *for every path* $\mathbf{p}'$ *that is a tail segment of* **p**, *it holds that* $\mathbf{p}'$ *is decided and* $\mathcal{B}(\mathbf{p}') = \mathcal{B}(\mathbf{p})$.

Proof. First assume the condition in the theorem is satisfied; we show that $\mathcal{B}$ is splittable. Take any branch **b**. Every branch is decided; therefore, **b** is decided and by Proposition 3.6 it has a core and **b** is head-determined at each position $i \geq \ell(core_{\mathcal{B}}(\mathbf{b}))$. The condition in the theorem now guarantees that **b** is tail-determined at each position $i < \ell(core_{\mathcal{B}}(\mathbf{b}))$ Thus, **b** is totally splittable. Since **b** was arbitrary, $\mathcal{B}$ is splittable indeed.

[9] A well-order is a total partial order $\langle S, \leq \rangle$ without infinite strictly descending sequences.

Now, assume $\mathcal{B}$ is splittable. Take an arbitrary core path $\mathbf{p} = x_0 \to \ldots \to x_k \to \ldots$ and an arbitrary tail path $\mathbf{p}' = x_k \to \ldots$ of $\mathbf{p}$. We need to show that $\mathbf{p}'$ is decided and $\mathcal{B}(\mathbf{p}') = \mathcal{B}(\mathbf{p})$. To prove this, it suffices to show that each branch $\mathbf{b}'$ extending $\mathbf{p}'$ has value $\mathcal{B}(\mathbf{p})$.

Take an arbitrary branch $\mathbf{b}'$ extending $\mathbf{p}'$. Its concatenation $\mathbf{b} = (x_0 \to \ldots \to x_k)\circ\mathbf{b}'$ is a branch and is totally splittable. This branch extends $\mathbf{p}$, therefore $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{p})$ (Proposition 3.3). Moreover, $\mathbf{b}$ is tail-determined at x_k (Theorem 3.7), hence $\mathcal{B}(\mathbf{b}') = \mathcal{B}(\mathbf{b})$. Together, it follows that $\mathcal{B}(\mathbf{b}') = \mathcal{B}(\mathbf{p})$. □

Proposition 2.21 guarantees that $\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{KK}}, \mathcal{B}_{\mathrm{st}}$ and $\mathcal{B}_{\mathrm{wf}}$ are splittable. The core paths of $\mathcal{B}_{\mathrm{sp}}$ are paths $x_0 \to x_1$, those of $\mathcal{B}_{\mathrm{KK}}$ and $\mathcal{B}_{\mathrm{wf}}$ are the branches, and those of $\mathcal{B}_{\mathrm{st}}$ are positive and negative branches, and paths of positive defined facts ending in a negative defined fact, and paths of negative facts ending in a positive defined fact.

The condition in Theorem 3.8 is a strong condition, but it has a natural explanation. A branch $x_0 \to x_1 \to x_2 \to \ldots$ formalizes a linear sequence "x_0 because x_1 because x_2 because ..." which goes on until the question can be decided. By its nature, such a "Why? Therefore!" sequence for x_0 comprises a sequence of the same nature for x_1, and then for x_2, etc., and the value of one is the value of the next. This motivates the constraint $\mathcal{B}(\mathbf{p}) = \mathcal{B}(\mathbf{p}')$ for all tails $\mathbf{p}'$ of core paths $\mathbf{p}$.

This explanation also suggests several other potentially interesting properties for branch evaluations. E.g., some branch evaluations go "all the way" with "why?" questions and do not stop asking "why?" in the middle of a branch. These are the tail-determined branch evaluations such as $\mathcal{B}_{\mathrm{KK}}$ and $\mathcal{B}_{\mathrm{wf}}$. Also, the process of asking "why?" has to end when an open fact o is reached, since such a fact is not explained by the justification frame. In this case, the natural answer for "why?" seems to be o itself. Thus, a natural property for a branch evaluation is that for a core path ending in an open fact o, the evaluation is o. This property is satisfied by all of $\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{KK}}, \mathcal{B}_{\mathrm{st}}$ and $\mathcal{B}_{\mathrm{wf}}$.

The tail-determined branch evaluations $\mathcal{B}_{\mathrm{KK}}$ and $\mathcal{B}_{\mathrm{wf}}$ differ in the way they evaluate positive, respectively negative loops, $\mathcal{B}_{\mathrm{KK}}$ mapping them to $\mathbf{u}$, $\mathcal{B}_{\mathrm{wf}}$ to $\mathbf{f}$, respectively $\mathbf{t}$. An intuitive explanation is as follows. According to $\mathcal{B}_{\mathrm{KK}}$, a fact x or $\sim x$ can be justified to be true only if it has an argument *from the ground up*. Infinite branches do not touch ground and are evaluated as $\mathbf{u}$. On the other hand, $\mathcal{B}_{\mathrm{wf}}$ incorporates an asymmetric vision on positive and negative facts. The default state of a positive fact is to be false; hence that of a negative fact is to be true. For a positive fact to be true, a constructive argument must exist. Infinite positive branches are not constructive and cannot be part of a good argument. Hence, any branch with a positive tail is evaluated to $\mathbf{f}$. On the other hand, a negative fact $\sim x$ can be false by default and does not need a constructive argument. To be true, it suffices that there is no constructive argument for x. A negative branch $\sim x \to \sim x_1 \to \sim x_2 \to \ldots$ should be read as: *there is no cause for x because there is no cause for x_1 because there is no cause for x_2 because ....* Such an infinite branch correctly reflects the absence of a constructive argument or cause for x and is given value $\mathbf{t}$.

We explore the properties of the concatenation operator $\circ$ on core paths.

Proposition 3.9. *Assume $\mathcal{B}$ is splittable. Let $\mathbf{p} = x_0 \to \ldots \to x_k, \mathbf{p}' = x_k \to \ldots$ be paths such that $\mathbf{p}$ is not decided and $\mathbf{p}'$ is core. Then $\mathbf{p}\circ\mathbf{p}'$ is core and $\mathcal{B}(\mathbf{p}\circ\mathbf{p}') = \mathcal{B}(\mathbf{p}')$.*

Proof. Take an arbitrary branch $\mathbf{b}$ extending $\mathbf{p}\circ\mathbf{p}'$. We need to prove: (1) $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{p}')$, and (2) all decided paths extended by $\mathbf{b}$ extend $\mathbf{p}\circ\mathbf{p}'$. Indeed, since the choice of $\mathbf{b}$ is arbitrary, we obtain from (1) that $\mathbf{p}\circ\mathbf{p}'$ is decided and from (2) that $\mathbf{p}\circ\mathbf{p}'$ does not extend a strictly smaller path that is decided.

The core path $\mathbf{p_b}$ of $\mathbf{b}$ strictly extends $x_0 \to \ldots \to x_k$ since the latter is not decided and neither is any of its initial segments. It holds that $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{p_b})$.

Since $\mathcal{B}$ is splittable, Theorem 3.8 guarantees that the tail path $\mathbf{p}_\mathbf{b}^k = x_k \to \ldots$ of $\mathbf{p_b}$ starting at x_k is decided and $\mathcal{B}(\mathbf{p_b}) = \mathcal{B}(\mathbf{p}_\mathbf{b}^k)$. Let $\mathbf{b}^k$ be the tail of $\mathbf{b}$ starting from position k. Both $\mathbf{p}'$ and $\mathbf{p}_\mathbf{b}^k$ are decided initial paths of $\mathbf{b}^k$. Since $\mathbf{p}'$ is core, $\mathbf{p}_\mathbf{b}^k$ extends $\mathbf{p}'$ and $\mathcal{B}(\mathbf{p}_\mathbf{b}^k) = \mathcal{B}(\mathbf{p}')$. Combining all equalities, we obtain (1) $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{p_b}) = \mathcal{B}(\mathbf{p}_\mathbf{b}^k) = \mathcal{B}(\mathbf{p}')$. Moreover, since every decided initial path of $\mathbf{b}$ extends $\mathbf{p_b}$ which extends $\mathbf{p}\circ\mathbf{p}'$, we also proved (2). □

Theorem 3.8 guarantees that the tail path of a core path is decided. The next proposition shows that, stronger, such a tail is core as well.

Proposition 3.10. *Let $\mathcal{B}$ be splittable. If $\mathbf{p}$ is core and $\mathbf{p}'$ is a tail path of $\mathbf{p}$, then $\mathbf{p}'$ is core.*

Proof. Let $\mathbf{p}'$ be the tail of $\mathbf{p}$ starting at position $k \geq 1$. Theorem 3.8 guarantees that $\mathbf{p}'$ is decided. Assume that $\mathbf{p}'$ extends a strictly smaller core path $\mathbf{p}''$. Then by Proposition 3.9, $(x_0 \to \ldots \to x_k)\circ\mathbf{p}''$ is a core path strictly extended by $\mathbf{p}$. This yields a contradiction. □

In summary, a tail of a core path is core, and vice versa, the concatenation of a non-decided path and a core path is a core path.

The notions of decided and core paths suggest to define a special class of partial justifications that are not locally complete but do contain enough information to be evaluated.

Definition 3.11. Let $\mathcal{B}$ be splittable. A justification J is decided (under $\mathcal{B}$) if every J-branch $\mathbf{p}$ is decided. A justification J is decided in x (under $\mathcal{B}$) if x is justified in J and every $\mathbf{p} \in B_J(x)$ is decided.

Decided justifications contain enough information to evaluate all their branches, and hence to derive the supported value of their justified facts. This allows us to extend the definition of supported value to decided justifications.

Definition 3.12. For each justified node x of a decided justification J, for each interpretation $\mathcal{I}$, we define

$$\sup_{\mathcal{B}}(x, J, \mathcal{I}) = \min_{\mathbf{p} \in B_J(x)} \mathcal{I}(\mathcal{B}(core_{\mathcal{B}}(\mathbf{p}))),$$

where min is with respect to $\leq_t$.

The following proposition shows that this extension is safe: the supported value of a justified fact in a decided justification equals the support in any justification that extends it.

Proposition 3.13. *For each justified node x of a decided justification J, for each locally complete extension K of J, for each interpretation $\mathcal{I}$, it holds that* $\sup_{\mathcal{B}}(x, K, \mathcal{I}) = \sup_{\mathcal{B}}(x, J, \mathcal{I})$.

Proof. Each branch $\mathbf{b}$ in K starting at x extends a core path $\mathbf{p}$ present in J. It holds that $\mathcal{B}(\mathbf{p}) = \mathcal{B}(\mathbf{b})$, and hence that $\mathcal{I}(\mathcal{B}(\mathbf{b})) = \mathcal{I}(\mathcal{B}(\mathbf{p}))$. □

Proposition 3.14. *A locally complete justification J is decided.*

Proof. Every J-branch is a $\mathcal{JF}$-branch, which is trivially decided. □

Definition 3.15. Let $\mathcal{B}$ be splittable. For every locally complete $\mathcal{JF}$-justification J justifying $x \in \mathcal{F}_d$, we define the core of J for x (under $\mathcal{B}$), denoted $core_{\mathcal{B}}(x, J)$, as the graph obtained from J by restricting its nodes and its edges to those that occur on core paths from x in J.

The sets of nodes and of edges on core paths in J are well-defined, therefore $core_{\mathcal{B}}(x, J)$ is a well-defined graph that exists and is unique. But is it a justification? Yes, as shown in the next proposition.

Proposition 3.16. *For every locally complete $\mathcal{JF}$-justification J justifying $x \in \mathcal{F}_d$, $J' = core_{\mathcal{B}}(x, J)$ is a proper, decided justification of x.*

Proof. By definition, $core_{\mathcal{B}}(x, J)$, contains only nodes on (core) paths reachable from x, so J' is proper.

If $y \to z$ is an edge of J', it occurs on a core path $\mathbf{p} = x \to \dots$. There is a rule $y \leftarrow B$ in $\mathcal{JF}$ such that B is the set of children of y in J. Since the initial segment $x \to \dots \to y$ is not core, it follows that all edges $y \to u$ in J occur on a core path from x and belong to J'. Hence, B is also the set of children of y in J'. It follows that J' is a justification.

Take any J'-branch $\mathbf{p}'$, take its first edge $y \to z$. There is a core path $x \to \dots \to y \to z \to \dots$ in J and in J', and its initial segment $\mathbf{p}_y = x \to \dots \to y$ is non-decided. Then $\mathbf{p} = \mathbf{p}_y \circ \mathbf{p}'$ is also a path of J and there is a branch $\mathbf{b}$ in J that extends $\mathbf{p}$. This branch has a core path which extends $\mathbf{p}_y$ and all its edges belong to J'. Its tail $y \to \dots$ is also core. Since $\mathbf{p}'$ is maximally long in J', $\mathbf{p}'$ extends this core path starting at y, hence $\mathbf{p}'$ is decided. □

Definition 3.17. We call a justification J a *core justification* of x if all its nodes and edges lie on core paths from x in J.

It follows that J is a core justification of x if and only if $J = core_{\mathcal{B}}(x, J)$. The core justifications of x are the minimal justifications that allow to assign a supported value to x in every interpretation. Every locally complete justification for x extends such a core justification. As such, one could say that core justifications are the building blocks of justifications.

Proposition 3.18. *Let $\mathcal{B}$ be a splittable branch evaluation, K a core justification for x and $\mathcal{I}$ an interpretation. For each y justified by K, it holds that* $\sup_{\mathcal{B}}(x, K, \mathcal{I}) \leq_t \sup_{\mathcal{B}}(y, K, \mathcal{I})$.

Proof. Since K is decided, every K-branch from y has a core. Every core path $\mathbf{p}'$ of y in K is a tail path of a core path $\mathbf{p}$ of x in K. Since $\mathcal{B}$ is splittable, $\mathcal{B}(\mathbf{p}')$ and $\mathcal{B}(\mathbf{p})$ are equal. As such the set of values $\mathcal{B}(\mathbf{p})$ of core paths $\mathbf{p}$ from y in K is a subset of the set of such values of core paths from x in K. The proposition now easily follows. □

The core justifications of x according to $\mathcal{B}_{\text{sp}}$ are the justifications of depth 1, corresponding to rules $x \leftarrow A$. The core justifications under $\mathcal{B}_{\text{KK}}$ and $\mathcal{B}_{\text{wf}}$ are proper locally complete justifications of x. The core justifications of x according to $\mathcal{B}_{\text{st}}$ are justifications where all justified facts have the same sign as x and all branches are infinite or end in an open fact or in a fact of the opposite sign.

Definition 3.19. A justification J offers *best support* for x in $\mathcal{I}$ if $\sup_{\mathcal{B}}(x, J, \mathcal{I}) = \text{SV}_{\mathcal{B}}(x, \mathcal{I})$.

Proposition 3.20. *Assume $\mathcal{B}$ is splittable. For each interpretation $\mathcal{I}$, for each defined fact x, there exists a core justification $J_{\mathcal{I}}^x$ of x that offers best support for x in $\mathcal{I}$.*

Proof. There exists a locally complete justification J such that $\mathrm{SV}_{\mathcal{B}}(x,\mathcal{I}) = \sup_{\mathcal{B}}(x,J,\mathcal{I})$. We can define $J_{\mathcal{I}}^x = core_{\mathcal{B}}(x,J)$ since Proposition 3.13 guarantees that $\sup_{\mathcal{B}}(x,J,\mathcal{I}) = \sup_{\mathcal{B}}(x,J_{\mathcal{I}}^x,\mathcal{I})$. □

3.2. Constructing complete justifications

As shown in Proposition 3.20, for every interpretation $\mathcal{I}$ and defined fact x there exists a core justification $J_{\mathcal{I}}^x$ of x that explains the supported value of x in $\mathcal{I}$, i.e., $\sup_{\mathcal{B}}(x,J_{\mathcal{I}}^x,\mathcal{I}) = \mathrm{SV}_{\mathcal{B}}(x,\mathcal{I})$. In this section, we investigate how to paste together such core justifications in one justification $J_{\mathcal{I}}$ that offers best support for *all* facts $x \in \mathcal{F}_d$ in $\mathcal{I}$. Such justifications will prove to be of tremendous use in the rest of the paper.

Definition 3.21. We call J a *complete support justification for $\mathcal{I}$* (under $\mathcal{B}$) if J is complete and moreover for every $x \in \mathcal{F}_d$, J offers best support in $\mathcal{I}$: $\sup_{\mathcal{B}}(x,J,\mathcal{I}) = \mathrm{SV}_{\mathcal{B}}(x,\mathcal{I})$.

The first step towards the construction of $J_{\mathcal{I}}$ is to define the operator to paste justifications together. We have defined a justification J as a graph $\langle N,E\rangle$. Recall that a justification J can also be viewed as a partial function from $\mathcal{F}_d$ to R with domain $\mathrm{dom}(J)$ the set of justified facts. The definition below uses the latter view.

Definition 3.22. For any two justifications J and K, we define $J\uparrow K$ as the unique justification with domain $\mathrm{dom}(J)\cup\mathrm{dom}(K)$ that extends J and coincides with K on $\mathrm{dom}(K)\setminus\mathrm{dom}(J)$.

Clearly, $J\uparrow K$ is an extension of J but it is only an extension of K in case the domains of J and K are disjoint. The following proposition shows that the pasting operation maps decided justifications to a decided one. Also, to some extent, the operation preserves the support value for justified facts.

Proposition 3.23. *Let J,K be decided justifications. The following claims hold.*

(a) *$J\uparrow K$ is a decided justification.*
(b) *For every interpretation $\mathcal{I}$, for every $x\in\mathrm{dom}(J)$, it holds that $\sup_{\mathcal{B}}(x,J,\mathcal{I}) = \sup_{\mathcal{B}}(x,J\uparrow K,\mathcal{I})$.*
(c) *For every $v\in\{\mathbf{t},\mathbf{u},\mathbf{f}\}$, if $\sup_{\mathcal{B}}(x,J,\mathcal{I})\geq v$ for every $x\in\mathrm{dom}(J)$, and $\sup_{\mathcal{B}}(y,K,\mathcal{I})\geq v$ for every $y\in\mathrm{dom}(K)$, then for every justified z of $J\uparrow K$, it satisfies $\sup_{\mathcal{B}}(z,J\uparrow K,\mathcal{I})\geq v$.*

Proof. (a) By construction, $L = J\uparrow K$ is a justification which justifies exactly the nodes $z\in\mathrm{dom}(J)\cup\mathrm{dom}(K)$. We need to prove that for any such z, each L-branch $\mathbf{p} = z\to\ldots$ extends a core path. First, assume $z\in dom(J)$. In that case, $\mathbf{p}$ is or extends a J-branch, which in turn extends a core path. Second, assume $z\in dom(K)$. There are two possibilities for $\mathbf{p}$. The first is that $\mathbf{p}$ extends a decided path $\mathbf{p}'$ in K which in turn extends a core path. The second is that $\mathbf{p}$ has an undecided initial path $\mathbf{p}_K := z\to\ldots\to u$ in K and then crosses into J at u. We can write $\mathbf{p}$ as $\mathbf{p}_K\circ\mathbf{p}_u$. Now, $\mathbf{p}_u$ extends a J-branch from u, which extends a core path $\mathbf{p}_u'$ from u. It follows from Proposition 3.9 that $\mathbf{p}_K\circ\mathbf{p}_u'$ is a core of $\mathbf{p}$. Thus, all L-branches extend some core path. It follows that L is decided. This finishes (a).

(b) follows from the fact that concatenation preserves all edges and core paths of J.

To show (c), we observe that every core path $\mathbf{p}$ in J and K has $\mathcal{I}(\mathcal{B}(\mathbf{p}))\geq v$. Furthermore, each core path in L is either a core of J or of K or it is a core path composed from an undecided initial segment in K and a core tail in L. Thus, all core paths $\mathbf{p}$ of L have $\mathcal{I}(\mathcal{B}(\mathbf{p}))\geq v$. (c) follows immediately. □

We now prove a main theorem of this section.

Theorem 3.24. *Let $\mathcal{JS}$ be justification system with splittable $\mathcal{B}$; let $\mathcal{I}$ be an interpretation. There exists a complete support justification $J_{\mathcal{I}}$ for $\mathcal{I}$. That is, for every $x\in\mathcal{F}_d$: $\sup_{\mathcal{B}}(x,J_{\mathcal{I}},\mathcal{I}) = \mathrm{SV}_{\mathcal{JS}}(x,\mathcal{I})$.*

Proof. We select for each $x\in\mathcal{F}_d$ a core justification $J_{\mathcal{I}}^x$ that provides best support for x in $\mathcal{I}$ (that is, $\sup_{\mathcal{B}}(x,J_{\mathcal{I}}^x,\mathcal{I}) = \mathrm{SV}_{\mathcal{B}}(x,\mathcal{I})$). By Proposition 3.20 such core justifications exist. To obtain $J_{\mathcal{I}}$, we paste all these justifications together in the way described next.

For each truth value $v\in\{\mathbf{t},\mathbf{u},\mathbf{f}\}$, by the well-ordering theorem,[10] there exists a (possibly transfinite) sequence that contains all facts x with $\mathrm{SV}_{\mathcal{B}}(x,\mathcal{I}) = v$. By composing these three sequences, we obtain a (possibly transfinite) sequence $\langle x_\alpha\rangle_{\alpha\geq 0}$ that contains every defined fact and in which the supported value of facts x_α in $\mathcal{I}$ decreases: if $\alpha\leq\beta$ then $\mathrm{SV}_{\mathcal{B}}(x_\alpha,\mathcal{I})\geq_t \mathrm{SV}_{\mathcal{B}}(x_\beta,\mathcal{I})$.

We construct the following sequence of justifications $\langle J_\alpha\rangle_{\alpha\geq 0}$, defined by transfinite induction:

[10] The well-ordering theorem is equivalent to the axiom of choice [32], which we assume in this paper.

1. J_0 is the empty justification.
2. $J_{\alpha+1} = J_\alpha \uparrow J_{\mathcal{I}}^{x_\alpha}$.
3. $J_\lambda = \cup_{\alpha<\lambda} J_\alpha$ for limit ordinal λ.

For each α in this sequence, we prove the following properties:

(a) J_α is a decided justification justifying at least all x_β, $\beta < \alpha$.
(b) J_α offers best support for all $x \in \mathrm{dom}(J_\alpha)$ in $\mathcal{I}$: $\sup_{\mathcal{B}}(x, J_\alpha, \mathcal{I}) = \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I})$.

The proof is by transfinite induction.

For the base case 0, J_0 trivially satisfies (a) and (b).

For the successor case $\alpha + 1$, assume J_α satisfies (a) and (b). According to Proposition 3.23(a), pasting decided justifications produces a decided justification. Hence, $J_{\alpha+1} = J_\alpha \uparrow J_{\mathcal{I}}^{x_\alpha}$ is decided and it obviously justifies at least every x_β, $\beta < \alpha + 1$. This proves (a) for $\alpha + 1$.

According to Proposition 3.23(b), J_α and $J_{\alpha+1}$ offer the same supported value for all facts $y \in \mathrm{dom}(J_\alpha)$ in all interpretations ($\sup_{\mathcal{B}}(y, J_\alpha, \mathcal{I}) = \sup_{\mathcal{B}}(y, J_{\alpha+1}, \mathcal{I})$), and J_α offers best support for these facts y in $\mathcal{I}$ ($\sup_{\mathcal{B}}(y, J_\alpha, \mathcal{I}) = \mathrm{SV}_{\mathcal{B}}(y, \mathcal{I})$). Hence, $J_{\alpha+1}$ offers best support in $\mathcal{I}$ for all $y \in \mathrm{dom}(J_\alpha)$. It remains to show the same for each $y \in \mathrm{dom}(J_{\mathcal{I}}^{x_\alpha}) \setminus \mathrm{dom}(J_\alpha)$.

Recall from Proposition 3.18, that each $y \in \mathrm{dom}(J_{\mathcal{I}}^{x_\alpha})$ satisfies $\sup_{\mathcal{B}}(y, J_{\mathcal{I}}^{x_\alpha}, \mathcal{I}) \geq \sup_{\mathcal{B}}(x_\alpha, J_{\mathcal{I}}^{x_\alpha}, \mathcal{I})(= \mathrm{SV}_{\mathcal{B}}(x_\alpha, \mathcal{I}))$, which entails $\mathrm{SV}_{\mathcal{B}}(y, \mathcal{I}) \geq_t \mathrm{SV}_{\mathcal{B}}(x_\alpha, \mathcal{I})$. If $\mathrm{SV}_{\mathcal{B}}(y, \mathcal{I}) >_t \mathrm{SV}_{\mathcal{B}}(x_\alpha, \mathcal{I})$, then y occurs earlier in the sequence than x_α and $y \in \mathrm{dom}(J_\alpha)$. Thus, any $y \in \mathrm{dom}(J_{\mathcal{I}}^{x_\alpha}) \setminus \mathrm{dom}(J_\alpha)$ has the same supported value as x_α in I. It follows from Proposition 3.23(c) that $\sup_{\mathcal{B}}(y, J_{\alpha+1}, \mathcal{I}) = \mathrm{SV}_{\mathcal{B}}(y, \mathcal{I})$. Thus, (b) holds for $\alpha + 1$.

The final case is for a limit ordinal λ. It is easy to see that $\mathcal{J}_\lambda = \cup_{\beta<\lambda} J_\beta$ is a justification with domain $\cup_{\beta<\lambda} \mathrm{dom}(J_\beta)$. For each y in its domain, there exists a least $\beta < \lambda$ such that J_β justifies y. This justification contains a core justification J_y for y which offers best support for y in $\mathcal{I}$ ($\sup_{\mathcal{B}}(y, J_y, \mathcal{I}) = \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I})$); J_y is present in all later justifications in the sequence, including in J_λ. Since this holds for every $y \in \mathrm{dom}(J_\lambda)$, it follows that J_λ is decided and offers best support for all its justified facts in $\mathcal{I}$.

We define $J_{\mathcal{I}}$ as the limit of the sequence. It justifies all defined facts and satisfies (a) and (b). Thus, J is a complete support justification for $\mathcal{I}$. □

The following theorem has already been proven in a slightly different context by Marynissen et al. [45]. It is a corollary of the previous Theorem 3.24 and Proposition 2.21.

Theorem 3.25. *Take $\mathcal{B} \in \{\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{KK}}, \mathcal{B}_{\mathrm{st}}, \mathcal{B}_{\mathrm{wf}}\}$. For every justification frame $\mathcal{JF}$ and every interpretation $\mathcal{I}$, there exists a complete support justification J for $\mathcal{I}$ under $\mathcal{B}$.*

3.3. Complementary justification frames and consistency

Several results in justification theory hold only for a particular class of justification frames, called *complementary frames*. This concept is related to the complementation operation defined in Definition 2.2. Recall that for a justification frame $\mathcal{JF}$, a selection function s for defined fact $x \in \mathcal{F}_d$ is a function $\mathcal{JF}(x) \to \mathcal{F}$ that maps the body A of a rule $x \leftarrow A$ to an element of A, denoted $s(A)$. We call $\sim x \leftarrow \sim \mathrm{Im}(s)$ the rule of s. Its body $\sim \mathrm{Im}(s)$ consists of the complement $\sim a$ of all facts a in the image of s. We say that a rule $x \leftarrow A$ subsumes a rule $x \leftarrow B$ if $A \subseteq B$, i.e., if A has fewer conditions than B.

Definition 3.26. A justification frame $\mathcal{JF}$ (or its rule set R) is *complementary* if for every $x \in \mathcal{F}_d$, (1) for every $x \leftarrow A \in R$, there exists a selection function s for $\sim x$ such that $\sim \mathrm{Im}(s) \subseteq A$ and (2) for every selection function s for $\sim x$ there exists a rule $x \leftarrow A$ such that $A \subseteq \sim \mathrm{Im}(s)$.

Thus, a justification frame is complementary if every rule $x \leftarrow A$ is subsumed by the rule of a selection function s of $\sim x$ and vice versa, every rule of a selection function s of $\sim x$ is subsumed by a rule $x \leftarrow A$.

Example 3.27. Consider the following rule set:

$$x \leftarrow a$$

$$x \leftarrow a, b$$

$$\sim x \leftarrow \sim a$$

The rule $\sim x \leftarrow \sim a$ is the rule of the selection function that selects a in both rules of x and subsumes the rule of the second selection function selecting a and b. The rule of the unique selection function of $\sim x$ is the first rule for x and subsumes the second one. Hence, this rule set (or the justification frame that contains it) is complementary. ▲

The following proposition shows that the complementation operation of Definition 2.2 can be used to produce complementary justification frames. It can be applied to create a complementary justification frame from a logic program, taking for F the set of atoms and R the set of rules of the program.

Proposition 3.28. *Let F be a subset of $\mathcal{F}_d$ such that $\{F, {\sim}F\}$ is a partition of $\mathcal{F}_d$. Let R be a rule set defining all and only facts of F. Then the complementation $R \cup R^*$ of R is complementary.*

Proof. Recall R^* is the set of rules of selection functions of facts $x \in F$ within the rule set R.

We observe that for facts $x \in {\sim}F$, $R \cup R^*$ satisfies conditions (1) and (2) by construction. Thus, it suffices to verify them for $x \in F$.

(1) Every $x \leftarrow A \in R$ is the rule of a selection function s for ${\sim}x$. Indeed, take the selection s for ${\sim}x$ that selects the negation of a fact of A in the body of every rule ${\sim}x \leftarrow B \in R^*$. The body of each such rule indeed contains the negation of a fact $a \in A$. Moreover, for every $a \in A$, there is at least one rule ${\sim}x \leftarrow B$ that contains ${\sim}a$ in its body. Hence ${\sim}\mathrm{Im}(s) = A$.

(2) Assume towards contradiction that there exists a rule of a selection function s for ${\sim}x$ in R^* that is not subsumed by some rule $x \leftarrow A \in R$. Then, for every rule $x \leftarrow A \in R$, there exists $a_A \in A \setminus {\sim}\mathrm{Im}(s)$. Consider the selection function s' for x that selects a_A in A, for every rule $x \leftarrow A \in R$. Then R^* contains the rule ${\sim}x \leftarrow {\sim}\mathrm{Im}(s')$ of s' and s selects some value ${\sim}a_A$ in that rule. This yields a contradiction. □

Proposition 3.29. *Let R be a complementary rule set. For each pair of rules $x \leftarrow A$ and ${\sim}x \leftarrow B$ in R, $A \cap {\sim}B \neq \emptyset$.*

Proof. Take $x \leftarrow A, {\sim}x \leftarrow B \in R$. By complementarity, there exists a selection function s of ${\sim}x$ such that ${\sim}\mathrm{Im}(s) \subseteq A$. Therefore, ${\sim}s(B) \in A$. On the other hand, $s(B) \in B$. □

This property has a useful extension to justifications.

Proposition 3.30. *Let R be a complementary rule set. Let J and K be locally complete justifications justifying x, respectively ${\sim}x$. There exists a J-branch $\mathbf{b}$ starting in x such that ${\sim}\mathbf{b}$ is a K-branch.*

Proof. This proposition follows by iterated application of Proposition 3.29. Indeed, the latter entails that there exist edges $x \to x_1$ in J and ${\sim}x \to {\sim}x_1$ in K. If $x_1 \in \mathcal{F}_o$, then both edges are branches in respectively J and K. If $x_1 \in \mathcal{F}_d$ then by repeating the argument there are edges $x_1 \to x_2$ in J and ${\sim}x_1 \to {\sim}x_2$ in K. By further iterating this argument, using transfinite induction, we obtain two branches $\mathbf{b}$ in J and ${\sim}\mathbf{b}$ in K. □

The following proposition was proven by Marynissen et al. [42].

Proposition 3.31. *Let $\mathcal{JS}$ be a complementary justification system such that $\mathcal{B}$ respects negation. For each $x \in \mathcal{F}_d$, for each interpretation $\mathcal{I}$, it holds that $\mathrm{SV}_{\mathcal{B}}({\sim}x, \mathcal{I}) \leq_t {\sim}\mathrm{SV}_{\mathcal{B}}(x, \mathcal{I})$.*

Proof. Let $v = \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I})$ and let J be a locally complete justification justifying x such that $\sup_{\mathcal{B}}(x, J, \mathcal{I}) = v$. Each branch $\mathbf{b}' = x \to \ldots$ in J satisfies $\mathcal{I}(\mathcal{B}(\mathbf{b}')) \geq_t v$. By Proposition 3.30, each justification K justifying ${\sim}x$ contains a branch $\mathbf{b} = {\sim}x \to \ldots$ such that ${\sim}\mathbf{b}$ is a branch of J. It follows that $\sup_{\mathcal{B}}({\sim}x, K, \mathcal{I}) \leq_t \mathcal{I}(\mathcal{B}(\mathbf{b})) = {\sim}\mathcal{I}(\mathcal{B}({\sim}\mathbf{b})) \leq_t {\sim}v$. Since this holds for every K, it follows that $\mathrm{SV}_{\mathcal{B}}({\sim}x, \mathcal{I}) \leq_t {\sim}v = {\sim}\mathrm{SV}_{\mathcal{B}}(x, \mathcal{I})$. □

Proposition 3.32. *Let $\mathcal{JS}$ be a complementary justification frame and x a defined fact. The following two conditions are equivalent.*

- *There exists a rule $x \leftarrow A \in R$ such that $\mathcal{I}(a) = \mathbf{t}$ for all $a \in A$.*
- *For all rules ${\sim}x \leftarrow B \in R$ there is an $a \in B$ such that $\mathcal{I}(a) = \mathbf{f}$.*

Proof. Let x be a defined fact. First let there be a rule $x \leftarrow A \in R$ such that all $a \in A$ are true in $\mathcal{I}$. Since R is complementary, Proposition 3.29 entails that every rule ${\sim}x \leftarrow B \in R$ contains a fact $a \in B$ such that ${\sim}a \in A$ and $\mathcal{I}(a) = \mathbf{f}$.

Vice versa, assume that for every ${\sim}x \leftarrow B \in R$, there exists an element $a \in B$ such that $\mathcal{I}(a) = \mathbf{f}$. Take the selection function s that selects these false a's from each such rule for ${\sim}x$. Since R is complementary, there is a rule $x \leftarrow A \in R$ that subsumes the rule of s, i.e., $A \subseteq {\sim}\mathrm{Im}(s)$. It follows that all facts of A are true in $\mathcal{I}$. □

For complementary justification frames, some useful properties hold. The next lemma gives a convenient way to prove that an interpretation is a model.

Lemma 3.33. *Take $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a complementary justification system and assume $\mathcal{B}$ respects negation. Every interpretation $\mathcal{I}$ such that $\mathrm{SV}_{\mathcal{JS}}(x, \mathcal{I}) \geq_t \mathcal{I}(x)$ for all $x \in \mathcal{F}_d$, is a $\mathcal{B}$-model of $\mathcal{JF}$.*

Proof. For all $x \in \mathcal{F}_d$ we have that $\mathrm{SV}_{\mathcal{JS}}(\sim x, \mathcal{I}) \geq_t \mathcal{I}(\sim x)$. It follows immediately from Proposition 3.31 that $\mathrm{SV}_{\mathcal{JS}}(\sim x, \mathcal{I}) \leq_t \sim \mathrm{SV}_{\mathcal{JS}}(x, \mathcal{I})$. Therefore, $\mathcal{I}(\sim x) \leq_t \sim \mathrm{SV}_{\mathcal{JS}}(x, \mathcal{I})$ or $\mathrm{SV}_{\mathcal{JS}}(x, \mathcal{I}) \leq_t \sim\mathcal{I}(\sim x) = \mathcal{I}(x)$. This completes the proof that $\mathrm{SV}_{\mathcal{JS}}(x, \mathcal{I}) = \mathcal{I}(x)$, i.e., $\mathcal{I}$ is a $\mathcal{B}$-model of $\mathcal{JF}$. □

One might expect that when a justification frame is complementary and $\mathcal{B}$ respects negation, the justification system is always guaranteed to be *consistent*, meaning that $\mathrm{SV}_{\mathcal{B}}(\sim x, \mathcal{I}) = \sim \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I})$. This is, however not the case and it remains an open question for which classes of branch evaluations consistency is guaranteed.[11] We refer the reader to the PhD of the first author [40] for details on this consistency question. What is important for the current paper is that for all branch evaluations considered here, consistency is indeed guaranteed.

Definition 3.34. A justification system $\mathcal{JS}$ is consistent if for every interpretation $\mathcal{I}$, for every $x \in \mathcal{F}_d$, $\mathrm{SV}_{\mathcal{B}}(\sim x, \mathcal{I}) = \sim \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I})$.

Theorem 3.35 *([40, Corollary 3.3.10]). Assume $\mathcal{JF}$ is complementary and $\mathcal{B}$ is one of the standard branch evaluations ($\mathcal{B} \in \{\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{st}}, \mathcal{B}_{\mathrm{wf}}, \mathcal{B}_{\mathrm{KK}}\}$). It holds that $\langle \mathcal{JF}, \mathcal{B}\rangle$ is consistent: for each $\mathcal{I}$,*

$$\mathrm{SV}_{\mathcal{B}}(\sim x, \mathcal{I}) = \sim \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I}).$$

3.4. Links between different justification models

Our next set of results is concerned with the relation between different semantics induced by JT. In the context of logic programming, it is well-known that there is a unique well-founded model, what the relation is between stable and well-founded models, or between the Kripke-Kleene and the well-founded model. Several such results will follow immediately by establishing the correspondence with AFT, but some of them will be needed in our proof. They are given explicitly, and sometimes in higher generality, in the current section.

First of all, in logic programming, it is well-known that the well-founded and Kripke-Kleene semantics induce a single model. In JT, we can prove that for each interpretation $\mathcal{I}_o$ of the open facts, there is a unique $\mathcal{B}_{\mathrm{KK}}$ and $\mathcal{B}_{\mathrm{wf}}$ model expanding $\mathcal{I}_o$. This result does not only hold for $\mathcal{B}_{\mathrm{KK}}$ and $\mathcal{B}_{\mathrm{wf}}$ but for the so-called class of *parametric* branch evaluations. A branch evaluation $\mathcal{B}$ is called *parametric* if $\mathcal{B}(\mathbf{b}) \in \mathcal{F}_o$ for all $\mathcal{JF}$-branches $\mathbf{b}$ and all justification frames $\mathcal{JF}$.

Proposition 3.36. *If $\mathcal{JF}$ is a justification frame and $\mathcal{B}$ a parametric branch evaluation, for each interpretation $\mathcal{I}_o$ of $\mathcal{F}_o$, $\mathcal{JF}$ has at most one $\mathcal{B}$-model $\mathcal{I}$ expanding $\mathcal{I}_o$. The model exists if and only if the system is consistent.*

Proof. The value of a justification with respect to a parametric branch evaluation only depends on the interpretation $\mathcal{I}_o$ of the open facts; hence $\sup_{\mathcal{B}}(x, J, \mathcal{I}) = \sup_{\mathcal{B}}(x, J, \mathcal{I}')$ for every two interpretations $\mathcal{I}$ and $\mathcal{I}'$ expanding $\mathcal{I}_o$. As such, $\sup_{\mathcal{B}}(x, J, \mathcal{I}_o)$ is well-defined.

If there exists a model $\mathcal{I}$ expanding $\mathcal{I}_o$, then it is the one such that $\mathcal{I}(x) = \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I}_o)$ for each $x \in \mathcal{F}_d$. This model is only well-defined if for all $x \in \mathcal{F}_d$, the supported values of x and $\sim x$ are consistent, that is, $\mathrm{SV}_{\mathcal{B}}(\sim x, \mathcal{I}_o) = \sim \mathrm{SV}_{\mathcal{B}}(x, \mathcal{I}_o)$. □

In other words, for any parametric $\mathcal{B}$, every interpretation $\mathcal{I}_o$ of the open facts induces at most one model. Theorem 3.35 guarantees that for $\mathcal{B}_{\mathrm{KK}}$ and $\mathcal{B}_{\mathrm{wf}}$ the model exists.

Corollary 3.37. *For every complementary justification frame, for every interpretation $\mathcal{I}_o$ of $\mathcal{F}_o$, there exists a unique $\mathcal{B}_{\mathrm{KK}}$-model and a unique $\mathcal{B}_{\mathrm{wf}}$-model expanding $\mathcal{I}_o$. When $\mathcal{F}_o = \emptyset$, then there is a unique $\mathcal{B}_{\mathrm{KK}}$-model and a unique $\mathcal{B}_{\mathrm{wf}}$-model.*

Similar to the situation in logic programming, the $\mathcal{B}_{\mathrm{KK}}$-model is the least precise $\mathcal{B}_{\mathrm{sp}}$-model expanding some $\mathcal{I}_o$. This is proven in the next two propositions.

Proposition 3.38. *The $\mathcal{B}_{\mathrm{KK}}$-model of a complementary justification system is also a $\mathcal{B}_{\mathrm{sp}}$-model.*

Proof. Let $\mathcal{I}$ be a $\mathcal{B}_{\mathrm{KK}}$-model of $\mathcal{JF}$. By Theorem 3.25, there is a complete support justification J for $\mathcal{I}$ under $\mathcal{B}_{\mathrm{KK}}$. The justification system is complementary and $\mathcal{B}_{\mathrm{sp}}$ respects negation, hence Lemma 3.33 applies for it. In particular, it suffices to prove that $\sup_{\mathcal{B}_{\mathrm{sp}}}(x, J, \mathcal{I}) \geq_t \mathcal{I}(x)$ for all $x \in \mathcal{F}_d$.

Since $\mathcal{B}_{\mathrm{KK}}$ is tail-determined, we have for every node y reachable from x in J that

$$\mathcal{I}(x) = \sup_{\mathcal{B}_{\mathrm{KK}}}(x, J, \mathcal{I}) \leq_t \sup_{\mathcal{B}_{\mathrm{KK}}}(y, J, \mathcal{I}) = \mathcal{I}(y).$$

The branch evaluation $\mathcal{B}_{\mathrm{sp}}$ maps a branch to its second node which is reachable from the start node. Therefore, $\sup_{\mathcal{B}_{\mathrm{sp}}}(x, J, \mathcal{I}) \geq_t \mathcal{I}(x)$, which concludes the proof. □

[11] However, for *tree-like* justifications, this question has recently been resolved [41].

In the following, the order $\leq_p$ is extended to interpretations, $\mathcal{I} \leq_p \mathcal{I}'$ if $\mathcal{I}(x) \leq_p \mathcal{I}'(x)$ for all $x \in \mathcal{F}$.

Proposition 3.39. *The $\mathcal{B}_{\text{KK}}$-model expanding $\mathcal{I}_o$ is the $\leq_p$-least $\mathcal{B}_{\text{sp}}$-model expanding $\mathcal{I}_o$.*

Proof. Let $\mathcal{I}_{\mathcal{B}_{\text{KK}}}$ be the (unique) Kripke-Kleene model expanding $\mathcal{I}_o$ and let $\mathcal{I}$ be any $\mathcal{B}_{\text{sp}}$-model expanding $\mathcal{I}_o$. We show that $\mathcal{I}_{\mathcal{B}_{\text{KK}}}(x) \leq_p \mathcal{I}(x)$ for all x. Take any fact x.

If $\mathcal{I}_{\mathcal{B}_{\text{KK}}}(x) = \mathbf{u}$, then clearly $\mathcal{I}_{\mathcal{B}_{\text{KK}}}(x) \leq_p \mathcal{I}(x)$, since $\mathbf{u}$ is the $\leq_p$-least truth value.

If $\mathcal{I}_{\mathcal{B}_{\text{KK}}}(x) = \mathbf{t}$, then $\text{SV}_{\mathcal{B}_{\text{KK}}}(x, \mathcal{I}) = \text{SV}_{\mathcal{B}_{\text{KK}}}(x, \mathcal{I}_{\mathcal{B}_{\text{KK}}}) = \mathbf{t}$. Let J be a proper justification for x such that $\sup_{\mathcal{B}_{\text{KK}}}(x, J, \mathcal{I}) = \mathbf{t}$. By definitions of $\mathcal{B}_{\text{KK}}$, this guarantees that J only has *finite* branches. We claim that for any node y in J, $\mathcal{I}(y) = \mathbf{t}$ and we prove this by induction on the depth of the subtree rooted in y. Clearly, the property holds for leaves (since $\sup_{\mathcal{B}_{\text{KK}}}(x, J, \mathcal{I})$ is $\mathbf{t}$ only if all leaves of J are true in $\mathcal{I}_o$). By induction, for other nodes, we know that the property holds for all nodes in $J(y)$ (which have a lower depth). Hence, $\sup_{\mathcal{B}_{\text{sp}}}(y, J, \mathcal{I}) = \mathbf{t}$. Since we assumed that $\mathcal{I}$ is a $\mathcal{B}_{\text{sp}}$-model, it must be that $\mathcal{I}(y) = \mathbf{t}$ indeed. In particular, the property holds for x and thus $\mathcal{I}(x) = \mathbf{t}$ as desired.

If $\mathcal{I}_{\mathcal{B}_{\text{KK}}}(x) = \mathbf{f}$, then $\mathcal{I}_{\mathcal{B}_{\text{KK}}}(\sim x) = \mathbf{t}$ and we can use the same argument as above to conclude that also $\mathcal{I}(\sim x) = \mathbf{t}$. $\square$

The relation between well-founded and stable models is similar: the $\mathcal{B}_{\text{wf}}$-model expanding an interpretation $\mathcal{I}_o$ of the open facts, is the least precise $\mathcal{B}_{\text{st}}$-model, as proven next.

Proposition 3.40. *The $\mathcal{B}_{\text{wf}}$-model of $\mathcal{JF}$ expanding $\mathcal{I}_o$ is a $\mathcal{B}_{\text{st}}$-model of $\mathcal{JF}$.*

Proof. Let $\mathcal{I}$ be the unique $\mathcal{B}_{\text{wf}}$-model of $\mathcal{JF}$ expanding $\mathcal{I}_o$. By Theorem 3.25, there is a complete support justification J under $\mathcal{B}_{\text{wf}}$. By Lemma 3.33, it suffices to prove that $\text{SV}_{\mathcal{B}_{\text{st}}}(x, J, \mathcal{I}) \geq_t \mathcal{I}(x)$ for all $x \in \mathcal{F}_d$.

For any internal node y reachable from x in J we have that $\mathcal{I}(x) \leq_t \mathcal{I}(y)$. Indeed, since $\mathcal{B}_{\text{wf}}$ is tail-determined, we have that for every $\mathbf{b} \in B_J(y)$ there is a branch $\mathbf{b}'$ in $B_J(x)$ so that $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{b}')$. This means that $\mathcal{I}(y) = \sup_{\mathcal{B}_{\text{wf}}}(y, J, \mathcal{I}) \geq_t \sup_{\mathcal{B}_{\text{wf}}}(x, J, \mathcal{I}) = \mathcal{I}(x)$.

For every $\mathbf{b}$ in $B_J(x)$ we have that $\mathcal{B}_{\text{st}}(\mathbf{b})$ is mapped to an element in $\mathbf{b}$ or that $\mathcal{B}_{\text{st}}(\mathbf{b}) = \mathcal{B}_{\text{wf}}(\mathbf{b})$. In both cases, we know that $\mathcal{I}(x) \leq_t \mathcal{I}(\mathcal{B}_{\text{st}}(\mathbf{b}))$. This means that $\mathcal{I}(x) \leq_t \sup_{\mathcal{B}_{\text{st}}}(x, J, \mathcal{I}) \leq_t \text{SV}_{\mathcal{B}_{\text{st}}}(x, \mathcal{I})$. This concludes the proof that $\mathcal{I}$ is a $\mathcal{B}_{\text{st}}$-model of $\mathcal{JF}$. $\square$

Lemma 3.41. *Let $\mathcal{I}$ be a $\mathcal{B}_{\text{st}}$-model. For every $x \in \mathcal{F}_d$, it holds that $\text{SV}_{\mathcal{B}_{\text{wf}}}(x, \mathcal{I}) \leq_p \mathcal{I}(x)$.*

Proof. By Theorem 3.25, there is a complete support justification J for $\mathcal{I}$ under $\mathcal{B}_{\text{st}}$.

Assume $\text{SV}_{\mathcal{B}_{\text{wf}}}(x, \mathcal{I}) = \mathbf{f}$. Since $\sup_{\mathcal{B}_{\text{wf}}}(x, J, \mathcal{I}) = \mathbf{f}$ this means that $B_J(x)$ contains a branch $\mathbf{b}$ with an (infinite) positive tail or ending in a false open fact. In any case, there are only finitely many internal sign switches $y_1, \ldots, y_n$ in $\mathbf{b}$, and let $y_0 = x$. It is easy to see that $\sup_{\mathcal{B}_{\text{st}}}(y_n, J, \mathcal{I}) = \mathbf{f}$ since $B_J(y_n)$ contains a positive branch or a finite branch (without sign switches) ending in a false open fact. For i with $0 \leq i < n$ we have that $\mathcal{I}(y_i) = \text{SV}_{\mathcal{B}_{\text{st}}}(y_i, \mathcal{I}) = \sup_{\mathcal{B}_{\text{st}}}(y_i, J, \mathcal{I}) \leq_t \mathcal{I}(y_{i+1})$. Therefore, we obtain that $\mathcal{I}(x) = \mathbf{f}$.

Second, assume $\text{SV}_{\mathcal{B}_{\text{wf}}}(x, \mathcal{I}) = \mathbf{t}$. By consistency of $\mathcal{B}_{\text{wf}}$ (Theorem 3.35), this is equivalent to

$$\text{SV}_{\mathcal{B}_{\text{wf}}}(\sim x, \mathcal{I}) = \mathbf{f},$$

which implies $\mathcal{I}(\sim x) = \mathbf{f}$ and $\mathcal{I}(x) = \mathbf{t}$.

Combining the two statements, we get that $\text{SV}_{\mathcal{B}_{\text{wf}}}(x, \mathcal{I}) \leq_p \mathcal{I}(x)$. $\square$

Proposition 3.42. *The $\mathcal{B}_{\text{wf}}$-model expanding $\mathcal{I}_o$ is the $\leq_p$-least $\mathcal{B}_{\text{st}}$-model expanding $\mathcal{I}_o$.*

Proof. Let $\mathcal{I}_{\mathcal{B}_{\text{wf}}}$ be the $\mathcal{B}_{\text{wf}}$-model and $\mathcal{I}$ a $\mathcal{B}_{\text{st}}$-model expanding $\mathcal{I}_o$. Lemma 3.41 entails for all $x \in \mathcal{F}_d$ that $\mathcal{I}_{\mathcal{B}_{\text{wf}}}(x) = \text{SV}_{\mathcal{B}_{\text{wf}}}(x, \mathcal{I}_o) \leq_p \mathcal{I}(x)$. Hence, $\mathcal{I}_{\mathcal{B}_{\text{wf}}} \leq_p \mathcal{I}$, for every $\mathcal{B}_{\text{st}}$-model. By Proposition 3.40, $\mathcal{I}_{\mathcal{B}_{\text{wf}}}$ is a $\mathcal{B}_{\text{st}}$-model, hence it is the $\leq_p$-least one. $\square$

Finally, it holds that $\mathcal{B}_{\text{st}}$-models are always supported, again, analogously to well-known results in logic programming.

Proposition 3.43. *Every stable model is a supported model.*

Proof. Take a $\mathcal{B}_{\text{st}}$-model $\mathcal{I}$. By Theorem 3.25, there is a complete support justification J for $\mathcal{I}$ under $\mathcal{B}_{\text{st}}$. Take any edge $x \to y$ in J. If we can prove that $\mathcal{I}(x) \leq_t \mathcal{I}(y)$ it will follow $\mathcal{I}(x) \leq_t \text{SV}_{\mathcal{B}_{\text{sp}}}(x, \mathcal{I})$. Applying Lemma 3.33, we obtain that $\mathcal{I}$ is a $\mathcal{B}_{\text{sp}}$-model.

We do a case analysis on y. First, assume y is an open fact or has different sign than x. Then $x \to y$ is the core of every J-branch $x \to y \to \ldots$ under $\mathcal{B}_{\text{st}}$. This means that $\mathcal{I}(x) \leq_t \mathcal{I}(\mathcal{B}_{\text{st}}(x \to y)) = \mathcal{I}(y)$. Second, assume x and y are defined facts of the same sign. Then for every $\mathbf{b} \in B_J(y)$, it holds that $\mathcal{B}_{\text{st}}(x \to \mathbf{b}) = \mathcal{B}_{\text{st}}(\mathbf{b})$. It follows that $\mathcal{I}(x) \leq \min_{\mathbf{b} \in B_J(y)} \mathcal{I}(\mathcal{B}_{\text{st}}(\mathbf{b})) = \mathcal{I}(y)$. In both cases, it holds that $\mathcal{I}(x) \leq_t \mathcal{I}(y)$. $\square$

4. Embedding JT in AFT

We now turn our attention to the main topic of this paper, namely formally proving the correspondence between JT and AFT. We start with a brief recall of the basic definitions that constitute AFT, next show how to obtain an approximator out of a justification frame, and finally prove that indeed, all major semantics are preserved under this correspondence.

4.1. Preliminaries: AFT

Given a complete lattice $\langle L, \leq \rangle$, Approximation Fixpoint Theory [21] uses the *bilattice* $L^2 = L \times L$. We define projection functions as usual: $(x,y)_1 = x$ and $(x,y)_2 = y$. Pairs $(x,y) \in L^2$ are used to approximate elements in the interval $[x,y] = \{z \mid x \leq z \leq y\}$. We call $(x,y) \in L^2$ *consistent* if $x \leq y$, i.e., if $[x,y]$ is not empty. The set of consistent pairs is denoted L^c. A pair (x,x) is called *exact* since it approximates only the element x. The *precision order* $\leq_p$ on L^2 is defined as $(x,y) \leq_p (u,v)$ if $x \leq u$ and $y \geq v$. If (x,y) and (u,v) are consistent, this means that $[u,v] \subseteq [x,y]$. If $\langle L, \leq \rangle$ is a complete lattice, then so is $\left\langle L^2, \leq_p \right\rangle$. AFT studies fixpoints of operators $O: L \to L$ through operators approximating O. An operator $A: L^2 \to L^2$ is an *approximator* of O if it is $\leq_p$-monotone and has the property that $A(x,x) = (O(x), O(x))$ for all $x \in L$. Approximators are internal in L^c (i.e., map L^c into L^c). We often restrict our attention to *symmetric* approximators: approximators A such that, for all x and y, $A(x,y)_1 = A(y,x)_2$. Denecker et al. [23] showed that the consistent fixpoints of interest of a symmetric approximator are uniquely determined by an approximator's restriction to L^c and hence, that it usually suffices to define approximators on L^c. Such a restriction is called a *consistent approximator*. As mentioned before, AFT studies fixpoints of O using fixpoints of A. The main type of fixpoints that concern us are given here.

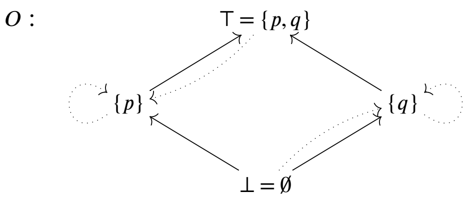
- A *partial supported fixpoint* of A is a fixpoint of A.
- The *Kripke-Kleene fixpoint* of A is the $\leq_p$-least fixpoint of A; it approximates all fixpoints of A.
- A *partial stable fixpoint* of A is a pair (x,y) such that $x = \mathrm{lfp}(A(\cdot,y)_1)$ and $y = \mathrm{lfp}(A(x,\cdot)_2)$, where $A(\cdot,y)_1$ denotes the function $L \to L: z \mapsto A(z,y)_1$ and analogously $A(x,\cdot)_2$ stands for $L \to L: z \mapsto A(x,z)_2$.
- The *well-founded fixpoint* of A is the $\leq_p$-least partial stable fixpoint of A.

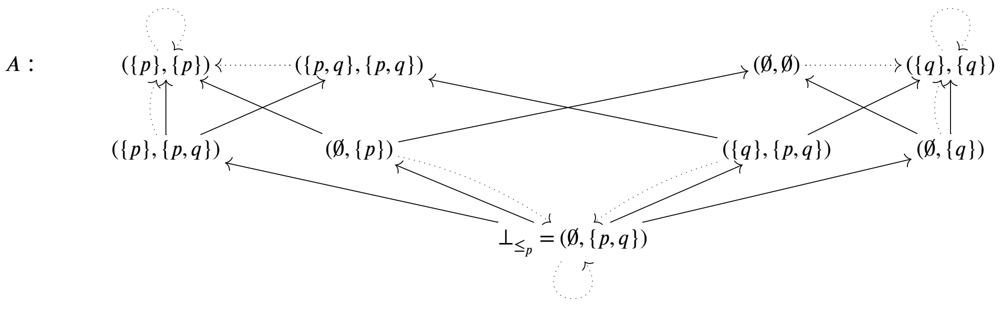
The adjective "partial" in this definition refers to "non-exact", in line with terminology used for instance in logic programming, where *partial stable models* are also defined. Uncoincidentally, partial stable models in logic programming correspond to what are called partial stable fixpoints here. Partial supported fixpoints of A are not necessarily fixpoints of O. However, in case they are exact, i.e., of the form (x,x), we call them *supported fixpoints*; in this case x is a fixpoint of O as well. We similarly define *stable fixpoints* as partial stable fixpoints that happen to be exact.

Example 4.1. As a small example[12] of the different constructions introduced here, we consider an operator and approximator as induced by the set of rules

$$\left\{ \begin{array}{l} p \leftarrow p \\ q \leftarrow \sim p. \end{array} \right\}.$$

The operator we use is the well-known immediate consequence operator of van Emden and Kowalski [51] and the approximator is Fitting's three-valued immediate consequence operator [29]; the lattices they operate on, and the operators are visualised below. Dashed arrows represent the order (on the lattice and the consistent part of the bilattice), and dotted arrows the operator and approximator.





[12] Many more examples can be found in the PhD thesis of Bogaerts [4].

We see that there are three (partial) supported fixpoints, namely $(\emptyset, \{p,q\})$ and $(\{p\},\{p\})$ and $(\{q\},\{q\})$. The latter two are exact and correspond to the two fixpoints of O. The former is not exact and is the Kripke-Kleene fixpoint. We claim that $(\{q\},\{q\})$ is a (partial) stable fixpoint (that happens to be exact). To see this we should verify (among others) that $\{q\} = \mathrm{lfp}(A(\cdot,\{q\})_1)$. This is indeed the case, since $A(\cdot,\{q\})_1$ maps $\emptyset$ to $\{q\}$ (indeed, $A(\emptyset,\{q\})_1 = (\{q\},\{q\})_1 = \{q\}$) and maps $\{q\}$ to itself.[13] In fact, $(\{q\},\{q\})$ is the only partial stable fixpoint since the other two partial supported fixpoints are not stable: $(\emptyset,\{p,q\})$ is not a stable fixpoint since $\mathrm{lfp}(A(\emptyset,\cdot)_2) = \{q\} \neq \{p,q\}$ and $(\{p\},\{p\})$ is not a stable fixpoint since $\mathrm{lfp}(A(\cdot,\{p\})_1) = \emptyset \neq \{p\}$. ▲

4.2. *The approximator*

Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ be a justification frame, fixed throughout this section. Our first goal is to define an approximator on a suitable lattice for $\mathcal{JF}$. Following the correspondence with how this is done in logic programming, we will take as lattice the set of *exact* interpretations (interpretations that map no facts to **u** except for **u** itself). It is easy to see that such interpretations correspond directly to subsets of $\mathcal{F}_+$. In other words, we will use the lattice $\langle L = 2^{\mathcal{F}_+}, \subseteq\rangle$. Now, the set L^c is isomorphic to the set of three-valued interpretations of $\mathcal{F}$; under this isomorphism, a consistent pair $(I,J) \in L^c$ corresponds to the three-valued interpretation $\mathcal{I}$ such that for positive facts $x \in \mathcal{F}_+$, $\mathcal{I}(x) = \mathbf{t}$ if $x \in I$, $\mathcal{I}(x) = \mathbf{f}$ if $x \notin$, and $\mathcal{I}(x) = \mathbf{u}$ otherwise.

Definition 4.2. The *operator* $O_{\mathcal{JF}} : L \to L$ of $\mathcal{JF}$ maps a subset I of $\mathcal{F}_+$ to

$$O_{\mathcal{JF}}(I) = \left\{x \in \mathcal{F}_+ \mid \exists x \leftarrow A \in R : \forall a \in A : (I,I)(a) = \mathbf{t}\right\}.$$

The *approximator* $A_{\mathcal{JF}} : L^c \to L^c$ of $\mathcal{JF}$ is defined as follows

$$A_{\mathcal{JF}}(\mathcal{I})_1 = \left\{x \in \mathcal{F}_+ \mid \exists x \leftarrow A \in R : \forall a \in A : \mathcal{I}(a) = \mathbf{t}\right\}$$
$$A_{\mathcal{JF}}(\mathcal{I})_2 = \left\{x \in \mathcal{F}_+ \mid \exists x \leftarrow A \in R : \forall a \in A : \mathcal{I}(a) \geq_t \mathbf{u}\right\}$$

Remark 4.3. This operator defined here is indeed well-defined: its image is in L^c, i.e., $A_{\mathcal{JF}}(\mathcal{I})_1 \leq A_{\mathcal{JF}}(\mathcal{I})_2$. To see this, note that whenever $\forall a \in A : \mathcal{I}(a) = \mathbf{t}$, clearly also $\forall a \in A : \mathcal{I}(a) \geq_t \mathbf{u}$. ▲

Proposition 4.4. *If no rule body in $\mathcal{JF}$ contains* **u**, *then $A_{\mathcal{JF}}$ is a consistent approximator of $O_{\mathcal{JF}}$.*

Proof. To see that it is $\leq_p$-monotonic, note that if $\mathcal{I}_1 \geq_p \mathcal{I}_2$, and $x \in A_{\mathcal{JF}}(\mathcal{I}_2)_1$, then there is a rule $x \leftarrow A$ such that $\forall a \in A : \mathcal{I}_2(a) = \mathbf{t}$. Since $\mathcal{I}_1 \geq_p \mathcal{I}_2$, then also $\forall a \in A : \mathcal{I}_1(a) = \mathbf{t}$ and hence $x \in A_{\mathcal{JF}}(\mathcal{I}_1)_1$ as well, i.e., $A_{\mathcal{JF}}(\mathcal{I}_2)_1 \subseteq A_{\mathcal{JF}}(\mathcal{I}_1)_1$ The argument for showing that $A_{\mathcal{JF}}(\mathcal{I}_1)_2 \subseteq A_{\mathcal{JF}}(\mathcal{I}_2)_2$ is similar. To see that $A_{\mathcal{JF}}$ coincides with $O_{\mathcal{JF}}$ on exact interpretations, we only need to show that $A_{\mathcal{JF}}(\mathcal{I})(x) \neq \mathbf{u}$ if $\mathcal{I}$ is exact. This follows directly from the fact that no rule body in $\mathcal{JF}$ contains **u**. □

The requirement that rule bodies cannot contain **u** is needed if we wish to follow the convention that approximators in AFT ought to be symmetric. Quoting Denecker et al. [23, page 14]:

> "While it is possible to develop a generalisation of the theory presented in this paper without the symmetry assumption, we chose to adopt it because the motivating examples, that is, operators occurring in knowledge representation, are symmetric."

Similarly, we are not aware of practical examples with bodies containing **u** in unnested justification systems.[14] However, in nested systems, as demonstrated by Marynissen [40, Chapter 6], bodies containing **u** can occur quite easily due to the construction of the so-called *compression*. No proof, except for Proposition 4.4, in this paper makes use of the fact that **u** does not appear in bodies of rules. This means that once consistent AFT is worked out for asymmetric operators, we can remove this restriction. In the rest of this paper, we assume that every justification frame does not have **u** in a rule body.

For a justification system, under certain conditions, we can derive a second operator on interpretations, based on the supported values under the branch evaluation of the system.

Definition 4.5. Let $\mathcal{JS}$ be a complementary justification system such that $\mathrm{SV}_{\mathcal{JS}}(\sim x, \mathcal{I}) = \sim \mathrm{SV}_{\mathcal{JS}}(x, \mathcal{I})$ for all x and $\mathcal{I}$. With $\mathcal{JS}$, we associate a *support operator* $\mathcal{S}_{\mathcal{JS}}$ on $\mathcal{F}$-interpretations. For any $\mathcal{F}$-interpretation $\mathcal{I}$, $\mathcal{S}_{\mathcal{JS}}(\mathcal{I})$ is the $\mathcal{F}$-interpretation that maps every x to $\mathrm{SV}_{\mathcal{JS}}(x, \mathcal{I})$. If $\mathcal{JS}$ consists of $\mathcal{JF}$ and $\mathcal{B}$, then we write $\mathcal{S}_{\langle \mathcal{JF}, \mathcal{B}\rangle}$ for $\mathcal{S}_{\mathcal{JS}}$.

As shown by Marynissen [40, Theorems 3.2.6 and 3.3.9], the condition on $\mathcal{JS}$ in Definition 4.5 is satisfied for all the major branch evaluations ($\mathcal{B}_{\mathrm{KK}}$, $\mathcal{B}_{\mathrm{st}}$, $\mathcal{B}_{\mathrm{sp}}$, and $\mathcal{B}_{\mathrm{wf}}$) whenever the justification frame is complementary, meaning that the support operator

[13] Readers familiar with logic programming semantics might have observed that $A(\cdot,\{q\})_1$ is actually the immediate consequence operator of the reduct [31] of the program with respect to $\{q\}$. This holds in general.

[14] You and Yuan [57] argue that **u** itself is needed only for a very special type of logic programs.

is well-defined for all the branch evaluations of interest in the current paper. Moreover, it turns out that in case our justification frame behaves well with respect to negation (if it is complementary), the approximator *equals* the support operator induced by the branch evaluation $\mathcal{B}_{\mathrm{sp}}$.

Lemma 4.6. *For a complementary justification frame $\mathcal{JF}$, the function $A_{\mathcal{JF}}$ and the support operator $\mathcal{S}_{\langle \mathcal{JF},\mathcal{B}_{\mathrm{sp}}\rangle}$ are equal.*

Proof. Take an interpretation $\mathcal{I}$. For any $x \in \mathcal{F}_+$, it is obvious that $A_{\mathcal{JF}}(\mathcal{I})(x) = \mathcal{S}_{\langle \mathcal{JF},\mathcal{B}_{\mathrm{sp}}\rangle}(\mathcal{I})(x)$. Take $x \in \mathcal{F}_-$. We have that $A_{\mathcal{JF}}(\mathcal{I})(x) = {\sim}A_{\mathcal{JF}}(\mathcal{I})({\sim}x) = {\sim}\mathcal{S}_{\langle \mathcal{JF},\mathcal{B}_{\mathrm{sp}}\rangle}(\mathcal{I})({\sim}x) = \mathcal{S}_{\langle \mathcal{JF},\mathcal{B}_{\mathrm{sp}}\rangle}(\mathcal{I})(x)$, where the first step follows since interpretations commute with negation and the last step is by the consistency of $\mathcal{S}_{\langle \mathcal{JF},\mathcal{B}_{\mathrm{sp}}\rangle}$ as shown by Marynissen et al. [45]. $\square$

4.3. Semantic correspondence

The central result of this section is the following theorem, which essentially states that for all major semantics, the branch evaluation in JT corresponds to the definitions of AFT.

Theorem 4.7. *Take a complementary justification frame $\mathcal{JF}$.*

1. *The partial supported fixpoints of $A_{\mathcal{JF}}$ are exactly the supported models of $\mathcal{JF}$.*
2. *The Kripke-Kleene fixpoint of $A_{\mathcal{JF}}$ is the unique Kripke-Kleene model of $\mathcal{JF}$.*
3. *The partial stable fixpoints of $A_{\mathcal{JF}}$ are exactly the stable models of $\mathcal{JF}$.*
4. *The well-founded fixpoint of $A_{\mathcal{JF}}$ is the unique well-founded model of $\mathcal{JF}$.*

These four points are proven independently; the first follows directly from our observation that $A_{\mathcal{JF}}$ and $\mathcal{S}_{\langle \mathcal{JF},\mathcal{B}_{\mathrm{sp}}\rangle}$ are in fact the same operator.

Proposition 4.8 *(Item 1 of Theorem 4.7). The partial supported fixpoints of $A_{\mathcal{JF}}$ are exactly the supported models of $\mathcal{JF}$.*

Proof. Follows directly from Lemma 4.6. $\square$

Recall from Proposition 3.39 that the $\mathcal{B}_{\mathrm{KK}}$-model of $\mathcal{JF}$ is the $\leq_p$-least $\mathcal{B}_{\mathrm{sp}}$-model. Given the correspondence of supported semantics of $A_{\mathcal{JF}}$ and $\mathcal{JF}$, the result for the Kripke-Kleene semantics follows.

Proposition 4.9 *(Item 2 of Theorem 4.7). The Kripke-Kleene fixpoint of $A_{\mathcal{JF}}$ is equal to the unique $\mathcal{B}_{\mathrm{KK}}$-model of $\mathcal{JF}$.*

Proof. This follows directly from Proposition 3.39 and Proposition 4.8 using the fact that the Kripke-Kleene fixpoint of $A_{\mathcal{JF}}$ is defined as the least fixpoint of $A_{\mathcal{JF}}$. $\square$

The proof of the third point of Theorem 4.7 is split in two parts, proven separately in the following propositions.

Proposition 4.10 *(Item 3 of Theorem 4.7; first direction). Each stable model of $\mathcal{JF}$ is a partial stable fixpoint of $A_{\mathcal{JF}}$.*

Proof. Let $\mathcal{I} = (I_1, I_2)$ be a $\mathcal{B}_{\mathrm{st}}$-model of $\mathcal{JF}$. We prove that $\mathrm{lfp}(A_{\mathcal{JF}}(\cdot, I_2)_1) = I_1$ and that $\mathrm{lfp}(A_{\mathcal{JF}}(I_1, \cdot)_2) = I_2$. By Proposition 3.43, it holds that $A_{\mathcal{JF}}(\mathcal{I}) = \mathcal{S}_{\langle \mathcal{JF},\mathcal{B}_{\mathrm{sp}}\rangle}(\mathcal{I}) = \mathcal{I}$. Therefore, we have that $A_{\mathcal{JF}}(I_1, I_2)_1 = I_1$ and $A_{\mathcal{JF}}(I_1, I_2)_2 = I_2$.

Take $I_1' \subsetneq I_1$ and assume by contradiction that $A_{\mathcal{JF}}(I_1', I_2)_1 = I_1'$. Define $\mathcal{I}' = (I_1', I_2)$. Therefore, $\mathcal{I}' <_p \mathcal{I}$ and $\mathcal{I}(x) = \mathbf{t}$ and $\mathcal{I}'(x) = \mathbf{u}$ for all $x \in I_1' \setminus I_1$. By Theorem 3.25, there is a complete support justification J for $\mathcal{I}$ under $\mathcal{B}_{\mathrm{st}}$. Define the partial order $\preceq_J$ on I_1: $y \preceq_J x$ if y is reachable in J from x through positive facts. Since J does not contain infinite positive branches starting from a fact $x \in I_1$, we have that $\preceq_J$ does not have infinitely descending chains; hence $\preceq_J$ is well-founded. The set $I_1 \setminus I_1'$ is not empty, hence has a minimal element x with respect to $\preceq_J$. Take a child y of x in J. We prove that $\mathcal{I}'(y) = \mathbf{t}$. If y is open, then $\mathbf{t} = \mathcal{I}(x) = \sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) \leq_t \mathcal{I}(y) = \mathcal{I}'(y)$. If y has a different sign than x, then $\mathbf{t} = \mathcal{I}(x) = \sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) \leq_t \mathcal{I}(y)$. This means that ${\sim}y \notin I_2$ since $y \in \mathcal{F}_-$; hence $\mathcal{I}'(y) = \mathbf{t}$. If y has the same sign as x, then $\mathbf{t} = \mathcal{I}(x) = \sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) \leq_t \sup_{\mathcal{B}_{\mathrm{st}}}(y, J, \mathcal{I}) = \mathcal{I}(y)$. Therefore, $y \in I_1$, which implies that $y \prec_J x$. This means that $y \notin I_1 \setminus I_1'$. We can conclude that $y \in I_1'$; hence $\mathcal{I}'(y) = \mathbf{t}$. This shows that $\sup_{\mathcal{B}_{\mathrm{sp}}}(x, J, \mathcal{I}') = \mathbf{t}$, hence $\mathrm{SV}_{\mathcal{B}_{\mathrm{sp}}}(x, \mathcal{I}') = \mathbf{t}$. This implies that $x \in A_{\mathcal{JF}}(I_1', I_2)_1 = I_1'$, which contradicts that $x \notin I_1'$; hence $I_1 = \mathrm{lfp}(A_{\mathcal{JF}}(\cdot, I_2)_1)$.

Take $I_1 \subseteq I_2' \subsetneq I_2$ and assume by contradiction that $A_{\mathcal{JF}}(I_1, I_2')_2 = I_2'$. Define $\mathcal{I}'' = (I_1, I_2')$. Therefore, $\mathcal{I} <_p \mathcal{I}''$ and $\mathcal{I}(x) = \mathbf{u}$ and $\mathcal{I}''(x) = \mathbf{f}$ for all $x \in I_2 \setminus I_2'$.

Define the partial order $\preceq_J'$ on I_2 the same as before. This order is also well-founded for a similar reason. The set $I_2 \setminus I_2'$ is not empty, hence has a minimal element x with respect to $\preceq_J'$. Take a child y of x in J. We prove that $\mathcal{I}''(y) \geq_t \mathbf{u}$. If y is open, then $\mathbf{u} = \mathcal{I}(x) = \sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) \leq_t \mathcal{I}(y) = \mathcal{I}''(y)$. If y has a different sign as x, then $\mathbf{u} = \mathcal{I}(x) = \sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) \leq_t \mathcal{I}(y)$. This means that $\sim y \notin I_2$ or $\sim y \in I_2 \setminus I_1$. Therefore, $\sim y \notin I'_2$ or $\sim y \in I'_2 \setminus I_1$, thus $\mathcal{I}''(y) \geq_t \mathbf{u}$. If y has the same sign as x, then $\mathbf{u} = \mathcal{I}(x) = \sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) \leq_t \sup_{\mathcal{B}_{\mathrm{st}}}(y, J, \mathcal{I}) = \mathcal{I}(y)$. Therefore, $y \in I_2$, which implies that $y <'_J x$. This means that $y \notin I_2 \setminus I'_2$, hence $y \in I'_2$. We conclude that $\mathcal{I}''(y) \geq_t \mathbf{u}$. This shows that $\sup_{\mathcal{B}_{\mathrm{sp}}}(x, J, \mathcal{I}'') \geq_t \mathbf{u}$, hence $\mathrm{SV}_{\mathcal{B}_{\mathrm{sp}}}(x, \mathcal{I}'') \geq_t \mathbf{u}$. This implies that $x \in A_{\mathcal{JF}}(I_1, I'_2)_2 = I'_2$, which contradicts that $x \notin I'_2$, concluding that $I_2 = \mathrm{lfp}(A_{\mathcal{JF}}(I_1, \cdot)_2)$. This finishes the proof that $\mathcal{I}$ is a partial stable fixpoint of $A_{\mathcal{JF}}$. □

For the other direction, we first need the following lemma.

Lemma 4.11. *Let $\mathcal{I}$ be a $\mathcal{B}_{\mathrm{sp}}$-model and $x \in \mathcal{F}_+$ with $\mathrm{SV}_{\mathcal{B}_{\mathrm{sp}}}(x, \mathcal{I}) = \mathbf{f}$. It holds that $\mathrm{SV}_{\mathcal{B}_{\mathrm{st}}}(x, \mathcal{I}) = \mathbf{f}$.*

Proof. Take an arbitrary justification J with x as internal node. There is a child y of x with $\mathcal{I}(y) = \mathbf{f}$. If y has a different sign than x, then $\sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) = \mathbf{f}$. Otherwise, by iterating this argument, we can construct either a finite branch ending in a z with $\mathcal{I}(z) = \mathbf{f}$ or an infinite branch such that every element z in $\mathbf{b}$ has $\mathcal{I}(z) = \mathbf{f}$ and z has the same sign as x. Since x is positive, this means that $\sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) = \mathbf{f}$. This concludes the proof that $\mathrm{SV}_{\mathcal{B}_{\mathrm{st}}}(x, \mathcal{I}) = \mathbf{f}$. □

We are now ready to prove the second part of the third point of Theorem 4.7.

Proposition 4.12 *(Item 3 of Theorem 4.7; second direction). Each partial stable fixpoint of $A_{\mathcal{JF}}$ is a stable model of $\mathcal{JF}$.*

Proof. Let $\mathcal{I} = (I_1, I_2)$ be a partial stable fixpoint of $A_{\mathcal{JF}}$. We prove that $\mathrm{SV}_{\mathcal{B}_{\mathrm{st}}}(x, \mathcal{I}) = \mathcal{I}(x)$ for $x \in \mathcal{F}_+$. By consistency of $\mathcal{S}_{\langle \mathcal{JF}, \mathcal{B}_{\mathrm{st}} \rangle}$ [45], this proves that $\mathcal{I}$ is a stable model of $\mathcal{JF}$. We prove our claim in three parts.

Part 1: $\mathrm{SV}_{\mathcal{B}_{\mathrm{st}}}(x, \mathcal{I}) = \mathcal{I}(x) = \mathbf{t}$ for all $x \in I_1$. Since I_1 is $\mathrm{lfp}(A_{\mathcal{JF}}(\cdot, I_2)_1)$, there is a sequence $(K_i)_{i \leq \beta}$ for some ordinal number β so that

- $K_0 = \emptyset$;
- $K_{i+1} = A_{\mathcal{JF}}(K_i, I_2)_1$ for all $i < \beta$;
- $K_\alpha = \bigcup_{i<\alpha} K_i$ for all limit ordinals $\alpha \leq \beta$;
- $K_\beta = I_1$.

Now, for every $x \in I_1$, there is a least ordinal i_x such that $x \notin K_{i_x}$, while $x \in K_{i_x+1}$. This means that there is a rule $x \leftarrow A_x$ such that for all $y \in A_x$ we have that $(K_{i_x}, I_2)(y) = \mathbf{t}$. Since $(K_{i_x}, I_2) \leq_p \mathcal{I}$ we have that $\mathcal{I}(y) = \mathbf{t}$ for all $y \in A_x$.

We now define J to be the justification with exactly the rules $x \leftarrow A_x$ for $x \in I_1$. Every J-branch is finite and ending in an element in $\mathcal{F}_- \cup \mathcal{F}_o$. Indeed, an infinite J-branch $x = x_0 \to x_1 \to \cdots$ implies the existence of a strictly decreasing sequence of ordinals $(i_{x_0}, i_{x_1}, \ldots)$, which cannot be the case. Since every J-branch is finite, and $\mathcal{B}_{\mathrm{st}}$ maps each finite branch to a fact occurring on that branch, the value of J-branch starting from x is an element of J, hence $\sup_{\mathcal{B}_{\mathrm{st}}}(x, J, \mathcal{I}) = \mathbf{t}$.

Part 2: $\mathrm{SV}_{\mathcal{B}_{\mathrm{st}}}(x, \mathcal{I}) = \mathcal{I}(x) = \mathbf{u}$ for all $x \in I_2 \setminus I_1$. Since I_2 is $\mathrm{lfp}(A_{\mathcal{JF}}(I_1, \cdot)_2)$, there is a sequence $(M_i)_{i \leq \beta}$ for some ordinal number β so that

- $M_0 = I_1$;
- $M_{i+1} = A_{\mathcal{JF}}(I_1, M_i)_2$ for $i < \beta$;
- $M_\alpha = \bigcup_{i<\alpha} M_i$ for limit ordinal $\alpha \leq \beta$;
- $M_\beta = I_2$.

For every $x \in I_2 \setminus I_1$, there is a least ordinal j_x such that $x \notin M_{j_x}$, while $x \in M_{j_x+1}$. Therefore, there is a rule $x \leftarrow C_x$ such that for all $y \in C_x$ we have that $(I_1, M_{j_x})(y) \geq_t \mathbf{u}$. Since $\mathcal{I} \leq_p (I_1, M_{j_x})$ we have that $\mathcal{I}(y) \geq_t \mathbf{u}$ for all $y \in C_x$.

Define J' to be the justification with exactly the rules $x \leftarrow C_x$ for $x \in I_2 \setminus I_1$. By a similar reasoning as in the first part, we have that every J'-branch is finite and ending in $I_1 \cup \mathcal{F}_- \cup \mathcal{F}_o$. Define J^* as $J' \uparrow J$, with J the justification from the first part. A J^*-branch is either a J'-branch or a concatenation of a J'-branch with a J-branch. This means that J^* does not have infinite branches. By construction, we have for every element y in J^* that $\mathcal{I}(y) \geq_t \mathbf{u}$. The evaluation of a J^*-branch is equal to an element in J^*; hence $\sup_{\mathcal{B}_{\mathrm{st}}}(x, J^*, \mathcal{I}) \geq_t \mathbf{u}$.

However, every justification for x has a branch $\mathbf{b}$ such that $\mathcal{I}(\mathcal{B}_{\mathrm{st}}(\mathbf{b})) \leq_t \mathbf{u}$; hence $\mathrm{SV}_{\mathcal{B}_{\mathrm{st}}}(x, \mathcal{I}) = \mathbf{u} = \mathcal{I}(x)$. Indeed, for every y with $\mathcal{I}(y) \leq_t \mathbf{u}$ and rule $y \leftarrow C$ there is a $c \in C$ with $\mathcal{I}(c) \leq_t \mathbf{u}$. This constructs a branch $\mathbf{b}$ such that for every element y in $\mathbf{b}$ we have that $\mathcal{I}(y) \leq_t \mathbf{u}$. We know that $\mathcal{B}_{\mathrm{st}}(\mathbf{b}) \neq \mathbf{t}$; otherwise $\mathbf{b}$ is completely negative or ending in $\mathbf{t}$. Therefore, $\mathcal{B}_{\mathrm{st}}$ maps $\mathbf{b}$ to an element in $\mathbf{b}$ or to $\mathbf{f}$ or $\mathbf{u}$. This proves that $\mathcal{I}(\mathcal{B}_{\mathrm{st}}(\mathbf{b})) \leq_t \mathbf{u}$.

Part 3: $\mathrm{SV}_{\mathcal{B}_{\mathrm{st}}}(x, \mathcal{I}) = \mathcal{I}(x) = \mathbf{f}$ for all $x \in \mathcal{F}_+ \setminus I_2$. This is immediate from Lemma 4.11. □

Example 4.13. Let $\mathcal{F} = \{x, \sim x, y, \sim y, z, \sim z\} \cup \mathcal{L}$, $\mathcal{F}_+ = \{x, y, z\}$, and let R be the (complementary) set of rules

$$\left\{\begin{array}{lll} x \leftarrow y & \sim x \leftarrow \sim y & \\ y \leftarrow \sim z & \sim y \leftarrow z & \\ z \leftarrow \sim x, \sim y & \sim z \leftarrow x & \sim z \leftarrow y \end{array}\right\}.$$

The approximator $A_{\mathcal{JF}}$ has three partial stable fixpoints, namely $(\{x, y\}, \{x, y\}), (\{z\}, \{z\})$ and $(\emptyset, \{x, y, z\})$.

Let us take a look at the fixpoint $(\{x, y\}, \{x, y\})$. Since it is a stable fixpoint, we know that $(\{x, y\}, \{x, y\})$ is a least fixpoint of $A_{\mathcal{JF}}(\cdot, \{x, y\})$. This operator is monotone with respect to $\subseteq$; hence we can construct the fixpoint by iteratively applying the operator on $(\emptyset, \{x, y\})$. This produces the following sequence.

$$(\emptyset, \{x, y\}) \to (\{y\}, \{x, y\}) \to (\{x, y\}, \{x, y\})$$

The first step uses the rule $y \leftarrow \sim z$, while the second step uses the rule $x \leftarrow y$. Combining the two we get the justification $x \to y \to \sim z$. All nodes of this justification are true in the model $(\{x, y\}, \{x, y\})$; all its internal nodes are positive, and each defined leaf is negative. This illustrates the first step of the proof of Proposition 4.12. By extending the found justification, we get a locally complete justification with the same value as the supported value. ▲

The last point now follows immediately.

Proposition 4.14 *(Item 4 of Theorem 4.7). The well-founded fixpoint of $A_{\mathcal{JF}}$ is the unique well-founded model of $\mathcal{JF}$.*

Proof. This follows directly by combining item 3 of Theorem 4.7 with Proposition 3.42. □

5. Ultimate semantics for justification frames

When applying AFT to new domains, there is not always a clear choice of approximator to use; the operator on the other hand is often clearer. Denecker et al. [23] studied the space of approximators and observed that consistent approximators can naturally be ordered according to their precision. Specifically, if A and B are approximators of O, we call A *more precise* than B if $A(x, y) \geq_p B(x, y)$ for all $(x, y) \in L^C$. It then holds that a more precise approximator yields more precise results: for instance, if A is more precise than B, then the A-well-founded fixpoint is guaranteed to be more precise than the B-well-founded fixpoint. They also observed that the space of consistent approximators of O has a most precise element, called the *ultimate approximator*, denoted $\mathrm{U}(O)$. This induces ultimate versions of the various AFT fixpoints.[15] In the context of logic programming, the step from the standard approximator (which is Fitting's partial immediate consequence operator [29]) to the ultimate approximator roughly boils down to using *supervaluations* [28,52] instead of Kleene's truth tables [36]. The ultimate approximator of an operator $O : L \to L$ has the following form [23]:

$$\mathrm{U}(O) : L^c \to L^c : (x, y) \mapsto \left(\bigwedge_{x \leq z \leq y} O(z), \bigvee_{x \leq z \leq y} O(z) \right),$$

where $\bigwedge$ (respectively $\bigvee$) are the greatest lower (respectively least upper) bound with respect to $\leq$. If an approximator A approximates an operator O, then we abuse notation by defining $\mathrm{U}(A) := \mathrm{U}(O)$.

In the context of justification theory, the justification frame uniquely determines the approximator at hand. Still, we show that it is possible to obtain ultimate semantics here as well. To do so, we will develop a method to transform a justification frame $\mathcal{JF}$ into its ultimate frame $\mathrm{U}(\mathcal{JF})$. We will then show that the approximator associated to $\mathrm{U}(\mathcal{JF})$ is indeed the ultimate approximator of $O_{\mathcal{JF}}$. The result is a generic mechanism to go from any semantics induced by justification theory (for arbitrary branch evaluations – not just for those that have an AFT counterpart) to an ultimate variant thereof. Our construction is as follows:

Definition 5.1. Let $\mathcal{JF}$ be a complementary justification frame. Let X be the set of rules with a positive head. Let X^* be the least (w.r.t. $\subseteq$) set containing X that is closed under the addition of rules $x \leftarrow A$

- if there is a rule $x \leftarrow B$ with $B \subseteq A$, or
- if there are rules $x \leftarrow \{y\} \cup A$ and $x \leftarrow \{\sim y\} \cup A$.

Let Y be the complementation of X^*. Then $\mathrm{U}(\mathcal{JF})$ is defined to be the complementary justification frame $\langle \mathcal{F}, \mathcal{F}_d, Y \rangle$. We call $\mathrm{U}(\mathcal{JF})$ the *ultimate frame* of $\mathcal{JF}$.

The first rule in the construction of X^* contains several redundant rules that would strictly speaking not be required. However, it is sometimes convenient to know that the rule set is closed under the first rule as well.

[15] This added precision has an increased computational cost [23, Theorems 6.12 and 6.13].

Example 5.2. Let $\mathcal{F}_d = \{x, \sim x\}$ and $\mathcal{F}_o = \{a, \sim a, b, \sim b\} \cup \mathcal{F}_o$. Take R to be the (complementary) set of rules

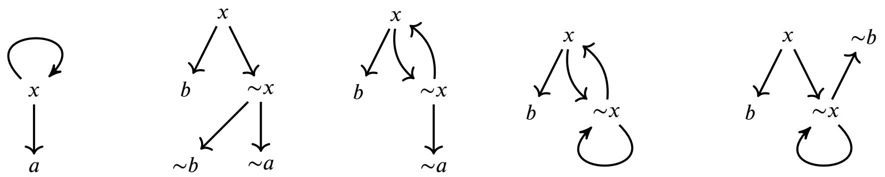
$$\left\{ \begin{array}{lll} x \leftarrow a, x & \sim x \leftarrow \sim a, \sim b & \sim x \leftarrow \sim a, x \\ x \leftarrow b, \sim x & \sim x \leftarrow \sim b, \sim x & \sim x \leftarrow x, \sim x \end{array} \right\}.$$

A rule $x \leftarrow A$ is minimal, if there is no rule $x \leftarrow B$ with $B \subset A$. For determining the supported value, one only needs to take minimal rules into account. The justification frame $\mathrm{U}(\mathcal{JF})$ has exactly the following minimal rules:

$$\begin{array}{ccc} x \leftarrow a, x & x \leftarrow b, \sim x & x \leftarrow a, b \\ \sim x \leftarrow \sim a, \sim b & \sim x \leftarrow \sim a, x & \sim x \leftarrow \sim b, \sim x \end{array}$$

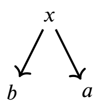
Of course, it contains many non-minimal rules, for example $x \leftarrow a, b, x$.

The justifications in the original system containing x as an internal node are exactly the following:



Assume from now on we are working under $\mathcal{B}_{\mathrm{st}}$. The value of the leftmost justification for x is $\mathbf{f}$ in every interpretation. The values of the other justifications for x are at most $\mathbf{u}$ in $\mathcal{B}_{\mathrm{st}}$-models. If it would be $\mathbf{t}$, then the value of these justifications for x is equal to the value of $\sim x$, which is $\mathbf{f}$.

In the ultimate justification frame, the minimal rule $x \leftarrow a, b$ is added and (as a consequence) on the negative side the minimal rule $\sim x \leftarrow x, \sim x$ is removed.[16] This allows for the justification



If the interpretation of a and b is $\mathbf{t}$, then the value of this justification for x is $\mathbf{t}$. Therefore, $(\{a, b, x\}, \{a, b, x\})$ is an ultimate stable model, while not a stable model. Note that the rightmost justification of $\mathcal{JF}$ is *not* a justification in $\mathrm{U}(\mathcal{JF})$, due to the removal of the rule $\sim x \leftarrow x, \sim x$. ▲

Remark 5.3. The ultimate semantics intuitively "joins" multiple rules for the same fact. That is, if there are two rules for p, say

$$p \leftarrow p \qquad p \leftarrow \sim p,$$

it treats these two as one monolithic rule

$$p \leftarrow (p \vee \sim p),$$

where it can reason powerfully over the propositional formula in the body. Of course, such disjunctive bodies cannot formally be expressed in justification theory. However, there are formalisms (such as *aggregates* [25] and *abstract constraint atoms* [38]) in which such complicated bodies can be represented. Moreover, ultimate semantics has been used to characterize and classify semantics of logic programs with aggregates [39,54]. ▲

It can be seen that the construction adds rules to $\mathcal{JF}$ in two cases. For the first type, if $x \leftarrow B$ is a rule in R with $B \subseteq A$, then if B is sufficient to derive x, clearly so is A. The second type of rule addition essentially performs some sort of case splitting. It states that if a set of facts A can be used with either y or $\sim y$ to derive x, then the essence for deriving x is the set A itself. In that case, the rule $x \leftarrow A$ is added to the ultimate frame. It turns out that this rule of case splitting is indeed sufficient to reconstruct the ultimate semantics in JT. This is formalised in the main theorem of this section:

Theorem 5.4. *For any frame $\mathcal{JF}$, $A_{\mathrm{U}(\mathcal{JF})} = \mathrm{U}(O_{\mathcal{JF}})$.*

An immediate corollary is, for instance that the set of stable models of $\mathrm{U}(\mathcal{JF})$ equals the set of ultimate stable fixpoints of $O_{\mathcal{JF}}$, and similarly for other semantics. Recall that, in the context of lattices with the subset order, which is what we are concerned with here, the ultimate approximator is defined as follows [23]:

[16] Technically, the definition of the ultimate frame does not say that this rule is "removed" but it recomputes the rules for negative facts by complementing X^*. The net effect is that this rule is removed.

$$\mathrm{U}(O)(I_1, I_2) = \left(\bigcap_{I_1 \subseteq K \subseteq I_2} O(K), \bigcup_{I_1 \subseteq K \subseteq I_2} O(K) \right). \tag{1}$$

The proof of Theorem 5.4 makes use of the following intermediate results.

Lemma 5.5. *Let $\mathcal{I}$ be an interpretation and $x \in \mathcal{F}_d$.*
If $O_{\mathcal{JF}}(\mathcal{I}')(x) = \mathbf{t}$ (respectively $\mathbf{f}$) for all exact interpretations $\mathcal{I}'$ with $\mathcal{I}' \geq_p \mathcal{I}$, then $A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})(x) = \mathbf{t}$ (resp. $\mathbf{f}$).

Proof. Let $\mathcal{I} = (I_1, I_2)$. We first prove this for the case $\mathbf{t}$. Define $X = I_2 \setminus I_1$, then we know that $X = \left\{ y \in \mathcal{F}_d \mid \mathcal{I}(y) = \mathbf{u} \right\}$. We prove for all $Y \subseteq X$ and all complete consistent subsets A over $X \setminus Y$ that the rule $x \leftarrow \{\mathbf{t}\} \cup I_1 \cup {\sim}(\mathcal{F}_+ \setminus I_2) \cup A$ is a rule in $\mathrm{U}(\mathcal{JF})$. A complete consistent subset A of Z is a subset such that for each $z \in Z$ exactly one of z and ${\sim}z$ is in A. If $Y = X$, then we get that $x \leftarrow \{\mathbf{t}\} \cup I_1 \cup {\sim}(\mathcal{F}_+ \setminus I_2)$ is a rule in $\mathrm{U}(\mathcal{JF})$ with every element y in its body we have $\mathcal{I}(y) = \mathbf{t}$, completing our proof. We prove our claim by well-founded induction.

Assume $Y = \emptyset$. Take a complete consistent set A over X. Define $K = I_1 \cup (A \cap \mathcal{F}_+)$. Since $\mathcal{I} \leq_p (K, K)$, we have a rule $x \leftarrow B$ in $\mathcal{JF}$ with $(K, K)(b) = \mathbf{t}$ for all $b \in B$. The true facts in (K, K) are $\{\mathbf{t}\} \cup I_1 \cup {\sim}(\mathcal{F}_+ \setminus I_2) \cup A$. So we have a rule $x \leftarrow B$ with $B \subseteq \{\mathbf{t}\} \cup I_1 \cup {\sim}(\mathcal{F}_+ \setminus I_2) \cup A$. By the construction of the ultimate fame (Definition 5.1), we can extend it to the rule $x \leftarrow I_1 \cup {\sim}(\mathcal{F}_+ \setminus I_2) \cup A$ in $\mathrm{U}(\mathcal{JF})$.

Take $Y \neq \emptyset$. Assume by induction the claim holds for all $Y' \subsetneq Y$. Take any $y \in Y$ and a complete consistent set A over $X \setminus Y$. Then $A \cup \{y\}$ and $A \cup \{{\sim}y\}$ are complete consistent sets over $X \setminus (Y \setminus \{y\})$. So by induction there are rules $x \leftarrow I_1 \cup (\mathcal{F}_+ \setminus I_2) \cup A \cup \{y\}$ and $x \leftarrow I_1 \cup (\mathcal{F}_+ \setminus I_2) \cup A \cup \{\sim y\}$ in $\mathrm{U}(\mathcal{JF})$. This means that $x \leftarrow I_1 \cup (\mathcal{F}_+ \setminus I_2) \cup A$ is a rule in $\mathrm{U}(\mathcal{JF})$.

The case for $\mathbf{f}$ follows easily from the $\mathbf{t}$ case. We have for all exact interpretations $\mathcal{I}'$ with $\mathcal{I} \leq_p \mathcal{I}'$ that $A_{\mathcal{JF}}(\mathcal{I}')(x) = \mathbf{f}$, i.e., by consistency that $A_{\mathcal{JF}}(I, I)({\sim}x) = \mathbf{t}$. By the first part of the lemma, we have that $A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})({\sim}x) = \mathbf{t}$; hence $A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})(x) = \mathbf{f}$. □

Combining this lemma with Eq. (1) of the ultimate approximator immediately yields that the operator $A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})$ is as least as precise as the ultimate approximator of $O_{\mathcal{JF}}$.

Lemma 5.6. *For all $\mathcal{I}$ we have $\mathrm{U}(O_{\mathcal{JF}})(\mathcal{I}) \leq_p A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})$.*

Proof. Take $x \in \mathcal{F}_+$.

If $\mathrm{U}(O_{\mathcal{JF}})(\mathcal{I})(x) = \mathbf{u}$, then it is obvious that $A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})(x) \geq_p \mathbf{u} = \mathrm{U}(O_{\mathcal{JF}})(\mathcal{I})(x)$.

If $\mathrm{U}(O_{\mathcal{JF}})(\mathcal{I})(x) = \mathbf{t}$, then $x \in \cap_{I_1 \subseteq K \subseteq I_2} O_{\mathcal{JF}}(K, K)_1$; hence $O_{\mathcal{JF}}(K, K)(x) = \mathbf{t}$ for all exact interpretations $(K, K) \geq_p \mathcal{I}$. Therefore, by Lemma 5.5, we have that $A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})(x) = \mathbf{t} = \mathrm{U}(O_{\mathcal{JF}})(\mathcal{I})(x)$.

If $\mathrm{U}(O_{\mathcal{JF}})(\mathcal{I})(x) = \mathbf{f}$, then $x \notin \cup_{I_1 \subseteq K \subseteq I_2} O_{\mathcal{JF}}(K, K)_1$; hence $O_{\mathcal{JF}}(K, K)(x) = \mathbf{f}$ for all exact interpretations $(K, K) \geq_p \mathcal{I}$. By Lemma 5.5 it follows that $A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})(x) = \mathbf{f} = \mathrm{U}(O_{\mathcal{JF}})(\mathcal{I})(x)$.

Similarly, we can prove for all $x \in \mathcal{F}_-$ that $\mathrm{U}(O_{\mathcal{JF}})(\mathcal{I})(x) \leq_p A_{\mathrm{U}(\mathcal{JF})}(\mathcal{I})(x)$. □

Since the ultimate approximator is the most precise approximator of any given operator, all that is left to prove, to indeed obtain Theorem 5.4 is that $A_{\mathrm{U}(\mathcal{JF})}$ indeed approximates $O_{\mathcal{JF}}$. That is the content of the last lemma.

Lemma 5.7. *$A_{\mathrm{U}(\mathcal{JF})}$ is an approximator of $O_{\mathcal{JF}}$.*

Proof. Take $I \subseteq \mathcal{F}_+$. Adding a rule $x \leftarrow B$ to $\mathcal{JF}$ if $x \leftarrow A$ is in $\mathcal{JF}$ with $A \subsetneq B$, does not change $A_{\mathcal{JF}}(I, I)$. Similarly, adding a rule $x \leftarrow A$ to $\mathcal{JF}$ if $x \leftarrow A \cup \{y\}$ and $x \leftarrow A \cup \{{\sim}y\}$ are in $\mathcal{JF}$ does not change $A_{\mathcal{JF}}(I, I)$. This means that $O_{\mathcal{JF}}(I) = A_{\mathcal{JF}}(I, I) = A_{\mathrm{U}(\mathcal{JF})}(I, I)$. □

6. Discussion and conclusion

In this paper, we presented a general mechanism to translate justification frames into approximating operators and showed that this transformation preserves all semantics the two formalisms have in common. The correspondence we established provides ample opportunity for future work and in fact probably generates more questions than it answers.

By embedding JT in AFT, JT gets access to a rich body of theoretical results developed for AFT. One example of this is *stratification*: we have a good understanding of when an approximator is *stratifiable* [55] and what this means for the computation of the fixpoints at hand (essentially, they can be computed "piece by piece" following the stratification). If we can now identify syntactic conditions on the justification frame at hand that guarantee that the induced operator is stratifiable, then we immediately know that those semantics that have a counterpart in AFT can indeed be computed following the stratification. Of course, there is a major limitation here: these results are only directly applicable to branch evaluations that have a counterpart in AFT. A question that immediately arises is whether results such as stratification results also apply to other branch evaluations, and which assumptions on branch evaluations would be required for that.

Another question that pops up on the justification side is whether concepts such as *groundedness* [10,11] or *safety* [12] can be transferred to JT.

On the AFT side, this embedding calls for a general algebraic study of *explanations*. Indeed, for certain approximators, namely those that are induced by a justification frame, our results give us a method for answering certain *why* questions in a graph-based manner (justifications allow us to answer why a fact is true or false in a given model). Lifting this notion of explanation to general approximators would benefit domains of logics that are covered by AFT but not by JT, such as auto-epistemic logic [46] and default logic [47]. A difficulty in doing so is that AFT is defined over arbitrary lattices, that do not necessarily need to be subset-lattices.

Another question that emerges naturally is how *nesting of justification frames* fits into this story. This notion of nesting was originally defined by Denecker et al. [19] in order to capture semantics of nested least and greatest fixpoints [33]. Denecker et al. defined the semantics of such nested justification systems using an operation called *compression*, which turns a nested system into an unnested one. Our embedding would in principle apply to the result of this operation, were it not that compression can result in rules with **u** in their body, thus violating the condition of Proposition 4.4. Recently, we have shown that the semantics of nested systems can also be characterized without compression, but using a different branch evaluation that makes a distinction between different levels of nesting [44]. Our embedding now raises the question whether this new branch evaluation gives rise to new fixpoint constructions or a notion of nested operator on the algebraic side.

Finally, justifications have appeared in many different forms in the literature. The formal connection between these different types of justifications has not yet been investigated. It would be interesting to find out if there are any formal connections, and to which extent the results of the current paper carry over to the framework of Fages [26], Schulz and Toni [48], or Cabalar et al. [15]. An important remark here is that these three frameworks all focus on providing justifications for a single semantics, whereas JT supports uniformly characterizing all major semantics of logic programming, through the use of different branch evaluations.

CRediT authorship contribution statement

Simon Marynissen: Formal analysis, Writing – original draft, Writing – review & editing. **Bart Bogaerts:** Conceptualization, Formal analysis, Funding acquisition, Project administration, Supervision, Writing – original draft, Writing – review & editing. **Marc Denecker:** Formal analysis, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] C. Antic, Fixed point semantics for stream reasoning, Artif. Intell. 288 (2020) 103370, https://doi.org/10.1016/j.artint.2020.103370.

[2] C. Antić, Neural logic programs and neural nets, https://hal.science/hal-04035402, 2024, preprint.

[3] Y. Bi, J. You, Z. Feng, A generalization of approximation fixpoint theory and application, in: R. Kontchakov, M. Mugnier (Eds.), Web Reasoning and Rule Systems - 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings, Springer, 2014, pp. 45–59.

[4] B. Bogaerts, Groundedness in logics with a fixpoint semantics, Ph.D. thesis, Department of Computer Science, KU Leuven, 2015, https://lirias.kuleuven.be/handle/123456789/496543, Denecker, Marc (supervisor), Vennekens, Joost and Van den Bussche, Jan (cosupervisors).

[5] B. Bogaerts, Weighted abstract dialectical frameworks through the lens of approximation fixpoint theory, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, the Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, the Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019.

[6] B. Bogaerts, L. Cruz-Filipe, Fixpoint semantics for active integrity constraints, Artif. Intell. 255 (2018) 43–70, https://doi.org/10.1016/j.artint.2017.11.003.

[7] B. Bogaerts, L. Cruz-Filipe, Stratification in approximation fixpoint theory and its application to active integrity constraints, ACM Trans. Comput. Log. 22 (2021) 6:1–6:19, https://doi.org/10.1145/3430750.

[8] B. Bogaerts, M. Jakubowski, Fixpoint semantics for recursive SHACL, in: A. Formisano, Y.A. Liu, B. Bogaerts, A. Brik, V. Dahl, C. Dodaro, P. Fodor, G.L. Pozzato, J. Vennekens, N. Zhou (Eds.), Proceedings 37th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2021, Porto (Virtual Event), 20-27th September 2021, 2021, pp. 41–47.

[9] B. Bogaerts, G. Van den Broeck, Knowledge compilation of logic programs using approximation fixpoint theory, Theory Pract. Log. Program. 15 (2015) 464–480, https://doi.org/10.1017/S1471068415000162, http://journals.cambridge.org/article_S1471068415000162.

[10] B. Bogaerts, J. Vennekens, M. Denecker, Grounded fixpoints and their applications in knowledge representation, Artif. Intell. 224 (2015) 51–71, https://doi.org/10.1016/j.artint.2015.03.006.

[11] B. Bogaerts, J. Vennekens, M. Denecker, Partial grounded fixpoints, in: Q. Yang, M. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 2784–2790, http://ijcai.org/papers15/Abstracts/IJCAI15-394.html.

[12] B. Bogaerts, J. Vennekens, M. Denecker, Safe inductions and their applications in knowledge representation, Artif. Intell. 259 (2018) 167–185, https://doi.org/10.1016/j.artint.2018.03.008, http://www.sciencedirect.com/science/article/pii/S000437021830122X.

[13] B. Bogaerts, J. Vennekens, M. Denecker, J. Van den Bussche, FO(C): a knowledge representation language of causality, Theory Pract. Log. Program. 14 (2014) 60–69, https://lirias.kuleuven.be/handle/123456789/459436.

[14] B. Bogaerts, A. Weinzierl, Exploiting justifications for lazy grounding of answer set programs, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 1737–1745.

[15] P. Cabalar, J. Fandinno, M. Fink, Causal graph justifications of logic programs, Theory Pract. Log. Program. 14 (2014) 603–618, https://doi.org/10.1017/S1471068414000234.

[16] A. Charalambidis, P. Rondogiannis, I. Symeonidou, Approximation fixpoint theory and the well-founded semantics of higher-order logic programs, Theory Pract. Log. Program. 18 (2018) 421–437, https://doi.org/10.1017/S1471068418000108.
[17] K.L. Clark, Negation as failure, in: Logic and Data Bases, Plenum Press, 1978, pp. 293–322.
[18] C.V. Damásio, A. Analyti, G. Antoniou, Justifications for logic programming, in: P. Cabalar, T.C. Son (Eds.), Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings, Springer, 2013, pp. 530–542.
[19] M. Denecker, G. Brewka, H. Strass, A formal theory of justifications, in: F. Calimeri, G. Ianni, M. Truszczyński (Eds.), Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings, Springer, 2015, pp. 250–264.
[20] M. Denecker, D. De Schreye, Justification semantics: a unifying framework for the semantics of logic programs, in: L.M. Pereira, A. Nerode (Eds.), LPNMR, MIT Press, 1993, pp. 365–379, https://lirias.kuleuven.be/handle/123456789/133075.
[21] M. Denecker, V. Marek, M. Truszczyński, Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning, in: J. Minker (Ed.), Logic-Based Artificial Intelligence, in: The Springer International Series in Engineering and Computer Science, vol. 597, Springer US, 2000, pp. 127–144.
[22] M. Denecker, V. Marek, M. Truszczyński, Uniform semantic treatment of default and autoepistemic logics, Artif. Intell. 143 (2003) 79–122, https://doi.org/10.1016/S0004-3702(02)00293-X.
[23] M. Denecker, V. Marek, M. Truszczyński, Ultimate approximation and its application in nonmonotonic knowledge representation systems, Inf. Comput. 192 (2004) 84–121, https://doi.org/10.1016/j.ic.2004.02.004, https://lirias.kuleuven.be/handle/123456789/124562.
[24] M. Denecker, J. Vennekens, Well-founded semantics and the algebraic theory of non-monotone inductive definitions, in: C. Baral, G. Brewka, J.S. Schlipf (Eds.), LPNMR, Springer, 2007, pp. 84–96.
[25] W. Faber, G. Pfeifer, N. Leone, Semantics and complexity of recursive aggregates in answer set programming, Artif. Intell. 175 (2011) 278–298, https://doi.org/10.1016/j.artint.2010.04.002.
[26] F. Fages, A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics, in: ICLP, MIT Press, 1990, p. 443.
[27] F. Fages, Consistency of Clark's completion and existence of stable models, Methods Log. Comput. Sci. 1 (1994) 51–60.
[28] M. Fitting, Tableaux for logic programming, J. Automat. Reason. 13 (1994) 175–188, https://doi.org/10.1007/BF00881954.
[29] M. Fitting, Fixpoint semantics for logic programming — a survey, Theor. Comput. Sci. 278 (2002) 25–51, https://doi.org/10.1016/S0304-3975(00)00330-3.
[30] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, On the implementation of weight constraint rules in conflict-driven ASP solvers, in: P.M. Hill, D.S. Warren (Eds.), ICLP, Springer, 2009, pp. 250–264.
[31] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R.A. Kowalski, K.A. Bowen (Eds.), ICLP/SLP, MIT Press, 1988, pp. 1070–1080, http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.6050.
[32] M. Hazewinkel, Encyclopaedia of Mathematics, Supplement III. Encyclopaedia of Mathematics, Springer Netherlands, 2007, https://books.google.be/books?id=47YC2h295JUC.
[33] P. Hou, B. De Cat, M. Denecker, FO(FD): extending classical logic with rule-based fixpoint definitions, Theory Pract. Log. Program. 10 (2010) 581–596, https://doi.org/10.1017/S1471068410000293.
[34] K. Inoue, C. Sakama, Negation as failure in the head, J. Log. Program. 35 (1998) 39–78, https://doi.org/10.1016/S0743-1066(97)10001-2.
[35] T. Janhunen, E. Oikarinen, H. Tompits, S. Woltran, Modularity aspects of disjunctive stable models, J. Artif. Intell. Res. 35 (2009) 813–857.
[36] S.C. Kleene, On notation for ordinal numbers, J. Symb. Log. 3 (1938) 150–155, http://www.jstor.org/stable/2267778.
[37] R. Lapauw, M. Bruynooghe, M. Denecker, Improving parity game solvers with justifications, in: D. Beyer, D. Zufferey (Eds.), Verification, Model Checking, and Abstract Interpretation - 21st International Conference, VMCAI 2020, New Orleans, LA, USA, January 16-21, 2020, Proceedings, Springer, 2020, pp. 449–470.
[38] V. Marek, M. Truszczyński, Logic programs with abstract constraint atoms, in: Proceedings of the 19th National Conference on Artificial Intelligence, AAAI-04, AAAI Press, 2004, pp. 86–91.
[39] M. Mariën, Model Generation for ID-Logic, Ph.D. thesis, Department of Computer Science, KU Leuven, Belgium, 2009.
[40] S. Marynissen, Advances in Justification Theory, Ph.D. thesis, Department of Computer Science, KU Leuven, 2022, https://lirias.kuleuven.be/3646147, Denecker, Marc and Bogaerts, Bart (supervisors).
[41] S. Marynissen, B. Bogaerts, Tree-like justification systems are consistent, in: Y. Lierler, J.F. Morales, C. Dodaro, V. Dahl, M. Gebser, T. Tekle (Eds.), Proceedings 38th International Conference on Logic Programming, ICLP 2022 Technical Communications / Doctoral Consortium, Haifa, Israel, 31st July 2022 - 6th, August 2022, 2022, pp. 1–11.
[42] S. Marynissen, B. Bogaerts, M. Denecker, Exploiting game theory for analysing justifications, Theory Pract. Log. Program. 20 (2020) 880–894, https://doi.org/10.1017/S1471068420000186.
[43] S. Marynissen, B. Bogaerts, M. Denecker, On the relation between approximation fixpoint theory and justification theory, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 1973–1980.
[44] S. Marynissen, J. Heyninck, B. Bogaerts, M. Denecker, On nested justification systems, Theory Pract. Log. Program. 22 (2022) 641–657, https://doi.org/10.1017/S1471068422000266.
[45] S. Marynissen, N. Passchyn, B. Bogaerts, M. Denecker, Consistency in justification theory, in: Proceedings of 17th International Workshop on Non-Monotonic Reasoning, NMR 2018, Tempe, Arizona, USA, Oct. 27-29, 2018, AAAI Press, 2018, pp. 41–52, http://www4.uma.pt/nmr2018/NMR2018Proceedings.pdf, 2018.
[46] R.C. Moore, Semantical considerations on nonmonotonic logic, Artif. Intell. 25 (1985) 75–94, https://doi.org/10.1016/0004-3702(85)90042-6.
[47] R. Reiter, A logic for default reasoning, Artif. Intell. 13 (1980) 81–132, https://doi.org/10.1016/0004-3702(80)90014-4.
[48] C. Schulz, F. Toni, ABA-based answer set justification, Theory Pract. Log. Program. 13 (2013).
[49] H. Strass, Approximating operators and semantics for abstract dialectical frameworks, Artif. Intell. 205 (2013) 39–70, https://doi.org/10.1016/j.artint.2013.09.004.
[50] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pac. J. Math. (1955).
[51] M.H. van Emden, R.A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (1976) 733–742, https://doi.org/10.1145/321978.321991.
[52] B. van Fraassen, Singular terms, truth-value gaps and free logic, J. Philos. 63 (1966) 481–495.
[53] P. Van Hertum, M. Cramer, B. Bogaerts, M. Denecker, Distributed autoepistemic logic and its application to access control, in: S. Kambhampati (Ed.), Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, IJCAI/AAAI Press, 2016, pp. 1286–1292, http://www.ijcai.org/Abstract/16/186.
[54] L. Vanbesien, M. Bruynooghe, M. Denecker, Analyzing semantics of aggregate answer set programming using approximation fixpoint theory, Theory Pract. Log. Program. 22 (2022) 523–537.
[55] J. Vennekens, D. Gilis, M. Denecker, Splitting an operator: algebraic modularity results for logics with fixpoint semantics, ACM Trans. Comput. Log. 7 (2006) 765–797, https://doi.org/10.1145/1182613.1189735.
[56] J. Vennekens, M. Mariën, J. Wittocx, M. Denecker, Predicate introduction for logics with a fixpoint semantics. Parts I and II, Fundam. Inform. 79 (2007) 187–227.
[57] J. You, L. Yuan, Three-valued formalization of logic programming: is it needed?, in: Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2-4, 1990, Nashville, Tennessee, USA, ACM Press, 1990, pp. 172–182, http://doi.acm.org/10.1145/298514.298559.