Tree-Like Justification Systems are Consistent*

Simon Marynissen
KU Leuven, Leuven, Belgium
Vrije Universiteit Brussel, Brussel, Belgium
simon.marynissen@kuleuven.be

Bart Bogaerts
Vrije Universiteit Brussel, Brussel, Belgium
bart.bogaerts@vub.be

Justification theory is an abstract unifying formalism that captures semantics of various non-monotonic logics. One intriguing problem that has received significant attention is the *consistency problem*: under which conditions are justifications for a fact and justifications for its negation suitably related. Two variants of justification theory exist: one in which justifications are trees and one in which they are graphs. In this work we resolve the consistency problem once and for all for the tree-like setting by showing that all reasonable tree-like justification systems are consistent.

1 Introduction

Justification theory [3] is a unifying theory to capture semantics of non-monotonic logics. Largely thanks to its abstract nature, it is a powerful framework with many use cases. First, it provides a mechanism to *define new logics* based on well-known principles in a uniform way, as well as to *transfer results* between domains. Second, it brings order in the zoo of logics and semantics, by enabling a *systematic comparison* between multiple semantics for a single logic and between different logics, for instance by answering the question whether a certain semantics of a given logic coincides with a semantics of another logic. Third, building on the notion of *nested justification systems*[1], it facilitates *modular definitions of knowledge representation languages and semantics*.

Justification theory builds on the semantic notion of a *justification*, which can intuitively be understood as an explanation (in the form of a tree or a directed graph) as to why a certain fact is true or false. For this reason, logics of which the semantics is captured by justification theory automatically come with a mechanism of *explanation*, which is of increasing importance for societal and legal reasons. On top of that, justifications have repeatedly proven to be useful algorithmically. They have been used in the unfounded set algorithm of Gebser et al. [5], for improving lazy grounding algorithms [1], as well as to speed-up parity game solvers [9].

The roots of justification theory can be traced back to the doctoral thesis of Denecker [2], where it was developed as a framework for studying semantics of logic programs. Later, Denecker et al. [3] developed a more general theory, aiming to also capture other knowledge representation formalisms, such as abstract argumentation [4], and nested least and greatest fixpoint definitions [8]. A notable difference between the original work of Denecker [2] and the theory of Denecker et al. [3] is that in the former work, a justification is a tree where the nodes are labeled with literals, while in the latter a justification is a (directed) graph. The relationship between these two formalisms was studied among others by Marynissen et al. [11]. For clarity, we will refer to *tree-like* and *graph-like* justifications when the distinction is important.

*This work was partially supported by Fonds Wetenschappelijk Onderzoek – Vlaanderen (project G0B2221N).

[1]Nested justification systems were originally defined by Denecker et al. [3], but have remained largely unexplored since then. In a companion paper to this paper [13], we provide a systematic study of nested systems and their properties

Submitted to:
ICLP 2022

© S. Marynissen & B. Bogaerts
This work is licensed under the
Creative Commons Attribution License.

In justification theory, whether or not a justification is "good" is determined by a *branch evaluation*. A branch evaluation is a function that associates to each path through a justification (i.e., to each branch of the justification) a value. However, a priori, it is not clear that each branch evaluation induces a well-defined semantics. For this to be the case, intuitively the justifications of a fact x and those of its negation $\sim x$ should be suitably related. Indeed, we cannot accept that there is both an explanation that x is true and an explanation that $\sim x$ is true. In general, to get a well-defined semantics we will need that the best possible justification for x and the best possible justification for $\sim x$ are complementary (either one of them is true and the other false, or both are unknown). The problem of determining whether or not a branch evaluation induces a well-defined semantics is known as the *consistency problem* and has been studied in several papers.

- Denecker [2, Theorem 4.3.1] studied the consistency problem for *tree-like* justifications for three specific branch evaluations (corresponding to completion semantics, stable semantics and well-founded semantics in logic programming).
- Marynissen et al. [14] were the first to exhibit a branch evaluation that (for graph-like justifications) is *not* consistent. Moreover, they showed that for *graph-like* justifications, four branch evaluations (in addition to the three mentioned above, also the Kripke-Kleene branch evaluation) are guaranteed to be consistent.
- Marynissen et al. [11] investigated the relationship between justification theory and games over graphs [7]. They used this relationship to identify some key properties that, when satisfied by a branch evaluation, guarantee that that branch evaluation is consistent for graph-like *and* tree-like justifications, but only in the context of a *finite* fact space.

Despite all these efforts, there is no clear understanding yet of what it is that makes a branch evaluation consistent. Moreover, proofs of consistency of individual branch evaluations often span several pages. In this paper, we resolve this question once and for all for tree-like justifications. Our main theorem states that: *for tree-like justifications, every reasonable branch evaluation is consistent*. The proof is surprisingly simple (compared to earlier proofs for individual branch evaluations) and is completely included in the main text. The results presented in this paper are part of the PhD thesis of the first author [10].

The rest of this paper is structured as follows. We recall some basic definitions of justification theory (focusing solely on tree-like justifications) in Section 2 and state the consistency problem in Section 3. In Section 4 we present the core theoretic results (in somewhat more generality) that are subsequently used to prove our main result (namely that every reasonable branch evaluation is consistent) in Section 5. We conclude in Section 6.

2 Preliminaries

We present the core definitions of justification theory, based on the formalization of Marynissen et al. [11, 12].

In the rest of this paper, let $\mathbb{F}$ be a set, referred to as a *fact space*, such that $\mathcal{L} = \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \subseteq \mathbb{F}$, where $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{u}$ have the respective meaning *true*, *false*, and *unknown*. The elements of $\mathbb{F}$ are called *facts*. The set $\mathcal{L}$ behaves as the three-valued logic with truth order $\leq_t$ given by $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$. We assume that $\mathbb{F}$ is equipped with an involution $\sim : \mathbb{F} \to \mathbb{F}$ (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 $x \neq \mathbf{u}$. For any fact x, $\sim x$ is called the *complement* of x. 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* $\mathbb{JF}$ is a tuple $\langle \mathbb{F}, \mathbb{F}_d, R \rangle$ such that

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

The set of *open* facts is denoted as $\mathbb{F}_o := \mathbb{F} \setminus \mathbb{F}_d$. 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 $\mathbb{JF}$ is denoted as $\mathbb{JF}(x)$. Rules $(x, A) \in R$ are often denoted as $x \leftarrow A$ and if $A = \{y_1, \ldots, y_n\}$, we often write $x \leftarrow y_1, \ldots, y_n$.

Logic programming rules can easily be transformed into rules in a justification frame. However, in logic programming, only rules for positive facts are given; never for negative facts.[2] Hence, in order to apply justification theory to logic programming, a mechanism for deriving rules for negative literals is needed as well. Similarly, in the setting of argumentation we can naturally derive rules for negative facts (given attack relations), but rules for positive facts are less obvious. For this, a technique called *complementation* was invented [3]; it is a generic mechanism that allows turning a set of rules for x into a set of rules for $\sim x$.

Complementation makes use of so-called selection functions for x. A *selection function* for x is a mapping $s\colon \mathbb{JF}(x) \to \mathbb{F}$ 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 of x. For a selection function s, the set $\{s(A) \mid A \in \mathbb{JF}(x)\}$ is denoted by $\mathrm{Im}(s)$. Selection functions can be used to construct rules for $\sim x$ from rules for x: intuitively, a fact $\sim x$ can be derived if every rule for x fails. Formally, if R is a set of rules, the complement $\mathrm{C}(R)$ is the set of elements of the form $\sim x \leftarrow \sim\mathrm{Im}(s)$ with x defined in R and s a selection function for x in R. The *complementation* of a set of rules R is $\mathrm{CC}(R) := R \cup \mathrm{C}(R)$.

In general, we will be interested in justification frames where the rules for x and the rules for $\sim x$ are suitably related. It does not necessarily have to be the case that our frame is obtained by complementation, but still we want to ensure that the rules are compatible. This has been studied intensively by Marynissen et al. [14], who have given several equivalent characterizations of when a justification frame is complementary. Here, we give only one such characterization:

Definition 2.2. Let $\mathbb{JF} = \langle \mathbb{F}, \mathbb{F}_d, R \rangle$ be a justification frame. We call $\mathbb{JF}$ *complementary* if for every $x \in \mathbb{F}_d$ the following hold:

1. for every selection function s for x in R, there exists an $A \in \mathbb{JF}(\sim x)$ with $A \subseteq \sim\mathrm{Im}(s)$;
2. for every $A \in \mathbb{JF}(x)$, there exists a selection function s for $\sim x$ in R with $\sim\mathrm{Im}(s) \subseteq A$.

The first item states that if we find a way to block every possible rule for x (by taking the complement of all the elements selected by a selection function s for x), then there should be a rule that derives $\sim x$, i.e., there should be some rule $\sim x \leftarrow A$ with A containing only facts in $\sim\mathrm{Im}(s)$. The second item states the

[2] In some extensions of logic programming "classical negation" is allowed in the head of rules. In that setting, an expression $\neg p$ is a shorthand for an arbitrary atom that can never be true together with p. This is significantly different from our setting where $\sim p$ simply means that p is false, and hence a rule with $\sim p$ in the head states a condition under which p is false. To further illustrate this difference: if there are no rules for $\sim p$, this means in our setting that $\sim p$ cannot be true (hence p cannot be false). In the aforementioned extensions of logic programming, a lack of rules for $\neg p$ entails nothing about p.

other direction, namely that whenever we can derive x (by means of a rule $x \leftarrow A$), it cannot be possible to $\sim x$, i.e., at least one fact in each rule for $\sim x$ should be blocked. This is expressed again by means of a selection function: there should exist a selection function that selects only facts that are the complement of a fact in A. For more intuition regarding complementarity, we refer the reader to Marynissen et al. [14, 11].

Example 2.3. The justification frame

$$\begin{aligned} p &\leftarrow q, \sim r \\ \sim p &\leftarrow \sim q \end{aligned}$$

is not complementary. Intuitively, the first rule states that p holds whenever q is true and r is false; this is the only case in which p can be derived. Since this is the only rule for p, we expect p to be false (i.e., $\sim p$ to be true) whenever q is false or r is true; the first case is present, but the second case for $\sim p$ is missing. After adding the rule

$$\sim p \leftarrow r \tag{1}$$

the frame becomes complementary. If we add a further rule

$$\sim p \leftarrow r, q, \tag{2}$$

the frame is still complementary; intuitively, (2) is a redundant rule, in the sense that it is weaker than (1). ▲

Definition 2.4. A *directed labeled graph* is a quadruple (N, L, E, ℓ) where N is a set of nodes, L is a set of labels, $E \subseteq N \times N$ is the set of edges, and $\ell \colon N \to L$ is a function called the labeling. An *internal* node is a node with outgoing edges and a *leaf* node is one without outgoing edges.

Definition 2.5. Let $\mathbb{JF} = \langle \mathbb{F}, \mathbb{F}_d, R \rangle$ be a justification frame. A *(tree-like) justification* J in $\mathbb{JF}$ is a directed labeled graph $(N, \mathbb{F}_d, E, \ell)$ such that

- the underlying undirected graph is a forest, i.e., is acyclic;
- for every internal node $n \in N$ it holds that $\ell(n) \leftarrow \{\ell(m) \mid (n,m) \in E\} \in R$.

Definition 2.6. A justification is *locally complete* if it has no leaves with label in $\mathbb{F}_d$. We call $x \in \mathbb{F}_d$ a *root* of a justification J if there is a node n labeled x such that every node is reachable from n in J.

We write $\mathfrak{J}(x)$ for the set of locally complete justifications rooted in a node labeled x.

Definition 2.7. Let $\mathbb{JF}$ be a justification frame. A $\mathbb{JF}$-*branch* is either an infinite sequence in $\mathbb{F}_d$ or a finite sequence in $\mathbb{F}_d$ followed by an element in $\mathbb{F}_o$. For a justification J in $\mathbb{JF}$, a J-*branch* starting in $x \in \mathbb{F}_d$ is a path in J starting in x that is either infinite or ends in a leaf of J. We write $B_J(x)$ to denote the set of J-branches starting in x.

Not all J-branches are $\mathbb{JF}$-branches since they can end in nodes with a defined fact as label. However, if J is locally complete, any J-branch is also a $\mathbb{JF}$-branch.

We denote a branch $\mathbf{b}$ as $\mathbf{b} : x_0 \to x_1 \to \cdots$ and define $\sim\mathbf{b}$ as $\sim x_0 \to \sim x_1 \to \cdots$.

Definition 2.8. A *branch evaluation* $\mathscr{B}$ is a mapping that maps any $\mathbb{JF}$-branch to an element in $\mathbb{F}$ for all justification frames $\mathbb{JF}$. A branch evaluation $\mathscr{B}$ *respects negation* if $\mathscr{B}(\sim\mathbf{b}) = \sim\mathscr{B}(\mathbf{b})$ for any branch $\mathbf{b}$. A justification frame $\mathbb{JF}$ together with a branch evaluation $\mathscr{B}$ forms a *justification system* $\mathbb{JS}$, which is presented as a quadruple $\langle \mathbb{F}, \mathbb{F}_d, R, \mathscr{B} \rangle$.

We now define some branch evaluations that induce semantics corresponding to the equally named semantics of logic programs.

Definition 2.9. The *supported* (completion) branch evaluation $\mathscr{B}_{\text{sp}}$ maps $x_0 \to x_1 \to \cdots$ to x_1 and branches that consist only of a singe open fact x_0 to x_0. The *Kripke-Kleene* branch evaluation $\mathscr{B}_{\text{KK}}$ maps finite branches to their last element and infinite branches to $\mathbf{u}$.

Other branch evaluations have been defined as well, for instance stable and well-founded branch evaluations, which correspond to the equally-named semantics of logic programming [6, 15]. We refer the reader to the original work introducing justification theory [3] for their definitions.

Definition 2.10. A *(three-valued) interpretation* of $\mathbb{F}$ is a function $I : \mathbb{F} \to \mathscr{L}$ such that $I(\sim x) = \sim I(x)$ and $I(\ell) = \ell$ for all $\ell \in \mathscr{L}$.

Definition 2.11. Let $\mathbb{JS} = \langle \mathbb{F}, \mathbb{F}_d, R, \mathscr{B} \rangle$ be a justification system, I an interpretation of $\mathbb{F}$, and J a locally complete justification in $\mathbb{JS}$. Let $x \in \mathbb{F}_d$ be a label of a node in J. The *value* of $x \in \mathbb{F}_d$ by J under I is defined as $\text{val}(J, x, I) = \min_{\mathbf{b} \in B_J(x)} I(\mathscr{B}(\mathbf{b}))$, where min is the minimum with respect to $\leq_t$.

The *supported value* of $x \in \mathbb{F}$ in $\mathbb{JS}$ under I is defined as

$$\text{SV}_{\mathbb{JS}}(x, I) = \max_{J \in \mathfrak{J}(x)} \text{val}(J, x, I) \text{ for } x \in \mathbb{F}_d$$

$$\text{SV}_{\mathbb{JS}}(x, I) = I(x) \text{ for } x \in \mathbb{F}_o.$$

In other words, the value of a fact in a justification is the value of the worst branch starting in that fact, and the supported value of a fact in an interpretation is the value of that fact in its best justification. Models are then defined as those interpretation in which the supported value of each fact equals their actual value. Every branch evaluation induces a different class of "models". For instance, models under the supported branch evaluation correspond to supported models in logic programming.

When $\mathbb{JS}$ is clear from the context, we will drop the subscript and just write $\text{SV}(x, I)$ for $\text{SV}_{\mathbb{JS}}(x, I)$.

Definition 2.12. Let $\mathbb{JS} = \langle \mathbb{F}, \mathbb{F}_d, R, \mathscr{B} \rangle$ be a justification system. An $\mathbb{F}$-interpretation I is a $\mathbb{JS}$-model if for all $x \in \mathbb{F}_d$, $\text{SV}(x, I) = I(x)$. If $\mathbb{JS}$ consists of $\mathbb{JF}$ and $\mathscr{B}$, then a $\mathbb{JS}$-model will also be called a $\mathscr{B}$-model of $\mathbb{JF}$.

3 The Consistency Problem

We defined *models* of a justification system by a kind of a fixpoint equation: for I to be a model, it must be a fixpoint of the operator $S_{\mathbb{JS}}$ that maps I to the interpretation $S_{\mathbb{JS}}(I)$, which is the function

$$\mathbb{F} \to \mathscr{L} : x \mapsto \text{SV}(x, I).$$

However, one intriguing problem is that domain and range of the support operator are not equal: the range of the support operator is the set of functions from $\mathbb{F}$ to $\mathscr{L}$, while the domain is the set of $\mathbb{F}$-interpretations, which are functions from $\mathbb{F}$ to $\mathscr{L}$ with some additional properties such as $I(\sim x) = \sim I(x)$ for all $x \in \mathbb{F}$. We might then wonder under which conditions, these properties will be guaranteed for $S_{\mathbb{JS}}(I)$.

On top of that, there is a more fundamental reason why this property is important. Explanations are of growing importance in various subdomains of artificial intelligence. In our setting, a justification with value $\mathbf{t}$ for x serves as an explanation of why x is true. But what if x is false? Which semantic structure can explain that? From the definition of supported value, it can be seen that x is false if *there* *are no justifications* with a better value (than $\mathbf{f}$) for x. But, the absence of such justifications is difficult to argue. The question that then remains is: how to show that there are no better justifications for x. The most obvious solution is considering a justification of $\sim x$. Indeed, intuitively, an explanation why the negation of x is true should explain why x is false. However, this method implicitly assumes that $\mathrm{SV}(\sim x, I) = \sim \mathrm{SV}(x, I)$, thus motivating the following definition.

Definition 3.1. A justification system $\mathbb{JS}$ is *consistent* if $\mathrm{SV}(\sim x, I) = \sim \mathrm{SV}(x, I)$ for every $x \in \mathbb{F}_d$ and every $\mathbb{F}$-interpretation I.

We are now ready to state the consistency problem, which is the central research question of this article.

Consistency problem

When is a justification system consistent? In particular, what properties do branch evaluations and justification frames need to have to ensure that the justification system is consistent?

Consistency is a reasonable assumption that, unfortunately, is not always satisfied. An obvious way to not satisfy it is by having unrelated rules for x and $\sim x$.

$$\left\{ \begin{array}{l} x \leftarrow \mathbf{t} \\ \sim x \leftarrow \mathbf{t} \end{array} \right\}$$

Of course, in this example we cannot expect that the justifications for x and $\sim x$ are related because their rules are contradictory. While justification theory in principle works with such rule sets, most of the theory has focused on complementary justification frames, where the rules for x and $\sim x$ are suitably related. Another way in which consistency can be violated is if $\mathscr{B}$ does not respect negation. Again, in its most general form, justification theory allows such strange branch evaluations (e.g., one that maps every branch to $\mathbf{t}$, in which every justification is always true), however there are no applications of such branch evaluations.

In fact, the main result of our paper is that for *tree-like* justifications, the two aforementioned syntactic properties are enough to guarantee consistency:

Theorem 3.2 (Main theorem)**.** *Let $\mathbb{JS} = \langle \mathbb{F}, \mathbb{F}_d, R, \mathscr{B} \rangle$ be a justification system. If $\mathbb{JS}$ is complementary and $\mathscr{B}$ respects negation, then $\mathbb{JS}$ is (tree-like) consistent.*

We do want to stress the fact that this only holds for tree-like justification systems: Marynissen et al. [14] gave an example of a branch evaluation that respects negation, on a complementary justification frame, that is *not* consistent for graph-like justifications (which we do not consider in this paper).

4 Constructing a Justification

In this section, we will provide the core theoretic results that are needed to prove our main theorem. In short, what we aim to show here is that if there are no "good" justifications for x, then we can construct a "good" justification for $\sim x$. We will prove this in some more generality, making use of a new concept: a *branch selection*: a set of branches starting in a certain fact x that contains at least one branch from every justification of x. The main result of this section (Theorem 4.5) then states that for every branch selection $\mathbb{B}$ for x, we can construct a justification for $\sim x$ that only has branches in $\sim\mathbb{B}$.

Throughout this section, we fix a justification system $\mathbb{JS} = \langle \mathbb{F}, \mathbb{F}_d, R, \mathscr{B} \rangle$ with $\mathbb{JF} = \langle \mathbb{F}, \mathbb{F}_d, R \rangle$.

Definition 4.1. A *branch selection* for x in $\mathbb{JF}$ is a set $\mathbb{B}$ of branches starting in x such that for each locally complete justification J rooted in x, $\mathbb{B}$ contains at least one J-branch.

In what follows, if $x \leftarrow A$ is a rule, and for each $y \in A$, J_y is a justification[3] rooted in y, we will write $x \xrightarrow{A} (J_y)_{y \in A}$ for the justification in which A are the children of x and such that the subtree rooted in $y \in A$ equals J_y.

Lemma 4.2. *Let $\mathbb{B}$ be a branch selection for x, and let $x \leftarrow A$ be a rule in $\mathbb{JF}$. There exists a $y^* \in A$ such that for every justification $J_{y^*} \in \mathfrak{J}(y^*)$ rooted in y^*, $\mathbb{B}$ contains at least one branch of the form $x \to y^* \to \mathbf{b}$ with $y^* \to \mathbf{b}$ a branch starting from the root of J_{y^*}.*[4]

Proof. Assume by contradiction that such a y^* does not exist. This would mean that for each $y \in A$, there exists a $J_y \in \mathfrak{J}(y)$ such that $\mathbb{B}$ does *not* contain any branches of the form $x \to y \to \mathbf{b}$ with $y \to \mathbf{b}$ a branch in J_y. However, if we have such a justification for each $y \in A$, consider the justification $x \xrightarrow{A} (J_y)_{y \in A}$. Since this is a locally complete justification of x, $\mathbb{B}$ must contain at least one of its branches, which yields a contradiction. $\square$

Corollary 4.3. *Let $\mathbb{B}$ be a branch selection for x, and let $x \leftarrow A$ be a rule in $\mathbb{JF}$. There exists a $y^* \in A$ such that $\mathbb{B}_{y^*} := \{y^* \to \mathbf{b} \mid x \to y^* \to \mathbf{b} \in \mathbb{B}\}$ is a branch selection for y^*.*

Proof. The only thing we need to show is that $\mathbb{B}_{y^*}$ contains at least one branch from each justification rooted in y^*. This follows directly from Lemma 4.2. $\square$

Corollary 4.4. *Assume $\mathbb{JS}$ is a complementary justification system. If $\mathbb{B}$ is a branch selection for x in $\mathbb{JF}$, then there exists a rule $\sim x \leftarrow B_{\sim x}$ for $\sim x$ such that for each $y \in \sim B_{\sim x}$, we have that $\mathbb{B}_y := \{y \to \mathbf{b} \mid x \to y \to \mathbf{b} \in \mathbb{B}\}$ is a branch selection for y.*

Proof. Corollary 4.3 determines a selection function for x in $\mathbb{JF}$, by choosing from each $x \leftarrow A$ a $y^* \in A$. Then by complementarity of $\mathbb{JS}$, there is a rule $\sim x \leftarrow B_{\sim x}$ such that every element of $B_{\sim x}$ is one of the selected y^*. $\square$

Theorem 4.5. *If $\mathbb{JS}$ is a complementary justification system and $\mathbb{B}$ is a branch selection for x in $\mathbb{JF}$, then there exists a justification J' rooted in $\sim x$ such that for each branch $\mathbf{b}'$ of J' starting in the root, $\sim \mathbf{b}' \in \mathbb{B}$.*

Proof. We will first construct a labeled tree T rooted in $\sim x$ with the properties that

1. the label of each internal node is a defined fact;
2. the label of each leaf is an open fact;
3. for each path in T starting in the root, the corresponing branch (obtained by taking the labels of each node) is the negation of a branch in $\mathbb{B}$.

However, our tree T will not necessarily be a justification: the set of children's labels of an internal node will not necessarily be the body of a rule for the label of that node. In the second part of the proof, we will then prove that a subtree of T indeed forms a justification.

Our tree T is constructed as follows.

- The **nodes** of T are all finite prefixes of the branches in $\sim\mathbb{B}$. That is, the nodes of T are all sequences $\sim x \to \sim x_1 \to \sim x_2 \to \cdots \to \sim x_n$ such that there is a branch in $\mathbb{B}$ that starts with $x \to x_1 \to \cdots \to x_n$. Each such node is **labeled** $\sim x_n$.

[3] In case y is an open fact, there is only one such justification, namely a justification with a single node and without edges.

[4] In case y^* is an open fact, this thus means that $\mathbf{b}$ is the empty branch and $y^* \to \mathbf{b}$ is a branch with only a single element y^*.

- For each prefix $\sim x \to \sim x_1 \to \sim x_2 \to \cdots \to \sim x_{n-1} \to \sim x_n$, there is an **edge** from $\sim x \to \sim x_1 \to \sim x_2 \to \cdots \to \sim x_{n-1}$ to $\sim x \to \sim x_1 \to \sim x_2 \to \cdots \to \sim x_n$.

By construction, there is a one-to-one correspondence between paths starting in the root of T and branches in $\mathbb{B}$. Since all branches in $\mathbb{B}$ are part of a locally complete justification, it is clear that indeed, the labels of internal nodes of this tree are defined facts and the labels of leaves are open facts.

We will now show that we can choose a subtree J' of T that is a locally complete justification. For each node $\eta = \sim x \to \sim x_1 \to \cdots \to \sim x_n$, we define $\mathbb{B}_\eta$ as the set of branches $\mathbf{b}$ starting in x_n such that $x \to x_1 \to \cdots \to x_{n-1} \to \mathbf{b}$ is a branch in $\mathbb{B}$. For some nodes, $\mathbb{B}_\eta$ is a branch selection function for x_n, but this is not guaranteed to be the case for every node. For *those nodes* η (with label $\sim z$) for which η *is* a branch selection for z, Corollary 4.4 guarantees there is a rule $\sim z \leftarrow B_{\sim z}$ such that for each $\sim y \in B_{\sim z}$, $\{y \to \mathbf{b} \mid z \to y \to \mathbf{b} \in \mathbb{B}_\eta\}$ is also a branch selection function for y. For each such η, we choose such a rule and for the rest of this proof denote $B_{\sim z}$ as $\mathit{chosenBody}(\eta)$. Given this choice, we construct the justification J inductively as follows:

- The root $\sim x$ is a node in J.
- For each node $\eta = \sim x \to \sim x_1 \to \sim x_n$ in J, and for each $\sim y \in \mathit{chosenBody}(\eta)$, the node $\sim x \to \sim x_1 \to \cdots \to \sim x_n \to \sim y$ of T is a node of J.

Of course, in order for this construction to work, we need to show that for each node in J, $\mathit{chosenBody}(\eta)$ is well-defined, i.e., that for each node η in J with label $\sim z$, $\mathbb{B}_\eta$ is a branch selection function for z. We prove this inductively as well

- This claim clearly holds for the root node since the corresponding branch selection function simply equals $\mathbb{B}$.
- If $\eta = \sim x \to \sim x_1 \to \cdots \to \sim x_n$ is a node for which $\mathbb{B}_\eta$ is a branch selection function for $x_n \in \mathbb{F}_d$ and $\sim y \in \mathit{chosenBody}(\eta)$, let η' denote the node $\sim x \to \sim x_1 \to \cdots \to \sim x_n \to \sim y$. We should show that $\mathbb{B}_{\eta'}$ is a branch selection for y. Now this follows easily from the fact that

$$\begin{aligned}\mathbb{B}_{\eta'} &= \{y \to \mathbf{b} \mid x \to x_1 \to \cdots \to x_n \to y \to \mathbf{b} \in \mathbb{B}\}\\ &= \{y \to \mathbf{b} \mid x_n \to y \to \mathbf{b} \in \mathbb{B}_\eta\}\end{aligned}$$

  and this last set is guaranteed to be a branch selection (by Corollary 4.4).

This way of constructing J indeed results in a locally complete justification: for each internal node η labeled $\sim z$ of T that is part of J, a rule $\sim z \leftarrow B$ is selected and the children of η have labels in B, which thus concludes our proof. □

5 Tree-Like Justification Systems are Consistent

We now turn our attention to proving the main theorem of this paper (Theorem 3.2). As before, we fix a justification system $\mathbb{JS} = \langle \mathbb{F}, \mathbb{F}_d, R, \mathscr{B} \rangle$ with $\mathbb{JF} = \langle \mathbb{F}, \mathbb{F}_d, R \rangle$.

One direction of the main theorem is fairly easy to prove: if we are given a good justification for a certain fact, there cannot be a good justification for its complement. This direction relies on the fact that for complementary frames, the cases of x and those of $\sim x$, and the justification of x and $\sim x$ are intrinsically related as formalized in the following two lemmas (inspired by similar results for graph-like justifications [14]).

Lemma 5.1. *If $\mathbb{JF}$ is complementary, then for all rules $x \leftarrow A$ and $\sim x \leftarrow B$ in R, we have $A \cap \sim B \neq \emptyset$.*

Proof. Take $A \in \mathbb{JF}(x)$ and $B \in \mathbb{JF}(\sim x)$. By complementarity, there exists a selection function s for $\sim x$ such that $\sim \mathrm{Im}(s) \subseteq A$. Therefore, $\sim s(B) \in A$. On the other hand, $s(B) \in B$; hence $\sim s(B) \in A \cap \sim B$. $\square$

Lemma 5.2. *Let $\mathbb{JF} = \langle \mathbb{F}, \mathbb{F}_d, R\rangle$ be a complementary justification frame and $x \in \mathbb{F}_d$. If J and K are justifications in $\mathfrak{J}(x)$ and $\mathfrak{J}(\sim x)$ respectively, then there exists a J-branch $\mathbf{b}$ starting in x such that $\sim\mathbf{b}$ is a K-branch starting in $\sim x$.*

Proof. We incrementally define J-paths $\mathbf{b}_i$ and K-paths $\mathbf{b}_i^*$ of length i such that $\mathbf{b}_i = \sim\mathbf{b}_i^*$. Define $\mathbf{b}_1$ and $\mathbf{b}_1^*$ as the node x and $\sim x$ respectively. Now assume that we obtained $\mathbf{b}_i$ and $\mathbf{b}_i^*$. Let y be the end node of $\mathbf{b}_i$. If y is not defined, so is $\sim y$ and then $\mathbf{b}_i$ is the desired J-branch. So assume y is defined. We want to find a fact z such that z is a child of y in J and $\sim z$ is a child of $\sim y$ in K. By using the rules for $y \leftarrow A$ in J and $\sim y \leftarrow B$ in K we can use Lemma 5.1 to obtain that $A \cap \sim B \neq \emptyset$ because $\mathbb{JF}$ is complementary. Choose a z in $A \cap \sim B$. Then we construct $\mathbf{b}_{i+1} = \mathbf{b}_i \to z$ and $\mathbf{b}_{i+1}^* = \mathbf{b}_i^* \to \sim z$. Our required branch is then the limit of $\mathbf{b}_i$ for i going to infinity. $\square$

From this lemma, one direction of consistency directly follows; this result was also already shown by Marynissen et al. [11, Proposition 5.13], but we include a direct proof to make the current paper self-contained.

Proposition 5.3. *If $\mathbb{JS}$ is complementary and $\mathscr{B}$ respects negation, then*

$$\mathrm{SV}(x,I) \leq_t \sim\mathrm{SV}(\sim x,I)$$

for any $x \in \mathbb{F}_d$ and any $\mathbb{F}$-interpretation I.

Proof. Take x with $\mathrm{SV}(x,I) = \ell$ for some $\ell \in \mathscr{L}$. This means there is a justification J such that $\mathrm{val}_{\mathbb{JS}}(J,x,I) = \ell$. Take a justification K for $\sim x$. Therefore, by Lemma 5.2, there is a J-branch $\mathbf{b}$ starting in x such that $\sim\mathbf{b}$ is a K-branch starting in $\sim x$. We consider three cases:

- If $\ell = \mathbf{t}$, then $I(\mathscr{B}(\mathbf{b})) = \mathbf{t}$; hence K has a branch that is evaluated to $\mathbf{f}$. Therefore, $\mathrm{val}_{\mathbb{JS}}(K,\sim x,I) = \mathbf{f}$. Since K was taken arbitrarily, we have that $\mathrm{SV}(\sim x,I) = \mathbf{f}$.
- If $\ell = \mathbf{u}$, we need to prove that $\mathbf{u} \geq_t \mathrm{SV}(\sim x,I)$. Similarly, every justification K for $\sim x$ has a branch $\sim\mathbf{b}$ starting in $\sim x$ such that $\mathbf{b}$ is a J-branch and $I(\mathscr{B}(\mathbf{b})) \geq_t \mathbf{u}$. This shows that $I(\mathscr{B}(\sim\mathbf{b})) \leq_t \mathbf{u}$. Therefore, $\mathrm{SV}(\sim x,I) \leq_t \mathbf{u}$.
- For $\ell = \mathbf{f}$, the statement is trivial. $\square$

The other direction of the consistency is completely novel and follows directly from the theory developed in Section 4.

Proposition 5.4. *If $\mathbb{JS}$ is complementary and $\mathscr{B}$ respects negation, then*

$$\mathrm{SV}(x,I) \geq_t \sim\mathrm{SV}(\sim x,I)$$

for any $x \in \mathbb{F}_d$ and any $\mathbb{F}$-interpretation I.

Proof. Consider the branch selection

$$\mathbb{B}_{\sim x} = \{\sim x \to \mathbf{b} \mid \mathscr{B}(\sim x \to \mathbf{b}) \leq_t \mathrm{SV}(\sim x,I)\}.$$

This set is a branch selection for $\sim x$, because each justification rooted in $\sim x$ must have at least one branch with a value at most $\mathrm{SV}(\sim x,I)$. Theorem 4.5 then guarantees that a justification J rooted in a node labelled x exists such that every branch $\mathbf{b}$ in J starting in the root is an element of $\sim\mathbb{B}_{\sim x}$; hence $\mathscr{B}(\sim\mathbf{b}) \leq_t \mathrm{SV}(\sim x, I)$. Since $\mathscr{B}$ respects negation, we have $\mathscr{B}(\mathbf{b}) \geq_t \sim\mathrm{SV}(\sim x, I)$ for every $\mathbf{b}$ in J starting in the root and thus that $\mathrm{val}(J,x,I) \geq_t \sim\mathrm{SV}(\sim x, I)$. From this, it immediately follows that $\mathrm{SV}(x,I) \geq_t \sim\mathrm{SV}(\sim x, I)$. □

Theorem 5.5 (Main theorem, restated)**.** *Let $\mathbb{JS} = \langle \mathbb{F}, \mathbb{F}_d, R, \mathscr{B} \rangle$ be a justification system. If $\mathbb{JS}$ is complementary and $\mathscr{B}$ respects negation, then $\mathbb{JS}$ is tree-like consistent.*

Proof. Follows by combining Proposition 5.3 with Proposition 5.4. □

6 Conclusion

Consistency is an important property that relates the absence of an explanation of a fact to the existence of an explanation for its complement. Our results do not have an impact on branch evaluations that have been used before, since for all of them, consistency has separately been proven. However, we believe it will simplify future applications of (tree-like) justification theory, by removing the burden of this proof obligation. Moreover, the general formulation of our results in Section 4, in fact entails that the so-called *flattening* of a justification system [10] is complementary. This notion of flattening is important in the context of *nested* justification systems [3, 13].

While we have now once and for all resolved the consistency question for tree-like justifications, for graph-like justifications, this problem is still open. However, in the PhD thesis of the first author [10, Corollary 2.4.7], we show that a justification system is graph-like consistent if and only if it is tree-like consistent and every tree-like justification can be transformed into a graph-like justification[5] (a property that is there called *graph-reducibility*). The results we presented in this work now entail that, under very mild assumptions, graph-like consistency is equivalent to graph-reducibility, and hence indirectly, also contributes to the understanding of graph-like justification systems.

References

[1] Bart Bogaerts & Antonius Weinzierl (2018): *Exploiting Justifications for Lazy Grounding of Answer Set Programs*. In Jérôme Lang, editor: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, ijcai.org, pp. 1737–1745, doi:10.24963/ijcai.2018/240. Available at `https://doi.org/10.24963/ijcai.2018/240`.

[2] Marc Denecker (1993): *Knowledge representation and reasoning in incomplete logic programming*. Ph.D. thesis, K.U.Leuven, Leuven, Belgium.

[3] Marc Denecker, Gerhard Brewka & Hannes Strass (2015): *A Formal Theory of Justifications*. In Francesco Calimeri, Giovambattista Ianni & Mirosław Truszczyński, editors: *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, *Lecture Notes in Computer Science* 9345, Springer, pp. 250–264, doi:10.1007/978-3-319-23264-5_22. Available at `http://dx.doi.org/10.1007/978-3-319-23264-5_22`.

[4] Phan Minh Dung (1995): *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games*. *Artif. Intell.* 77(2), pp. 321 – 357, doi:10.1016/0004-3702(94)00041-X. Available at `http://dx.doi.org/10.1016/0004-3702(94)00041-X`.

[5] In graph-like justifications, each fact occurs as the label of at most one node. The difficulty in this transformation is dealing with tree-like justifications that make multiple choices for a given fact.

[5] Martin Gebser, Roland Kaminski, Benjamin Kaufmann & Torsten Schaub (2009): *On the Implementation of Weight Constraint Rules in Conflict-Driven ASP Solvers*. In Patricia M. Hill & David Scott Warren, editors: *ICLP*, *LNCS* 5649, Springer, pp. 250–264. Available at `http://dx.doi.org/10.1007/978-3-642-02846-5`.

[6] Michael Gelfond & Vladimir Lifschitz (1988): *The Stable Model Semantics for Logic Programming*. In Robert A. Kowalski & Kenneth A. Bowen, editors: *ICLP/SLP*, MIT Press, pp. 1070–1080. Available at `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.6050`.

[7] Hugo Gimbert & Wieslaw Zielonka (2005): *Games Where You Can Play Optimally Without Any Memory*. In Martín Abadi & Luca de Alfaro, editors: *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, *Lecture Notes in Computer Science* 3653, Springer, pp. 428–442, doi:10.1007/11539452_33. Available at `https://doi.org/10.1007/11539452_33`.

[8] Ping Hou, Broes De Cat & Marc Denecker (2010): *FO(FD): Extending classical logic with rule-based fixpoint definitions*. *TPLP* 10(4-6), pp. 581–596. Available at `http://dx.doi.org/10.1017/S1471068410000293`.

[9] Ruben Lapauw, Maurice Bruynooghe & Marc Denecker (2020): *Improving Parity Game Solvers with Justifications*. In Dirk Beyer & Damien Zufferey, editors: *Verification, Model Checking, and Abstract Interpretation - 21st International Conference, VMCAI 2020, New Orleans, LA, USA, January 16-21, 2020, Proceedings*, *Lecture Notes in Computer Science* 11990, Springer, pp. 449–470, doi:10.1007/978-3-030-39322-9_21. Available at `https://doi.org/10.1007/978-3-030-39322-9_21`.

[10] Simon Marynissen (2022): *Advances in Justification Theory*. Ph.D. thesis, Department of Computer Science, KU Leuven. Available at `https://lirias.kuleuven.be/3646147`. Denecker, Marc and Bart Bogaerts (supervisors).

[11] Simon Marynissen, Bart Bogaerts & Marc Denecker (2020): *Exploiting Game Theory for Analysing Justifications*. *Theory Pract. Log. Program.* 20(6), pp. 880–894, doi:10.1017/S1471068420000186. Available at `https://doi.org/10.1017/S1471068420000186`.

[12] Simon Marynissen, Bart Bogaerts & Marc Denecker (2021): *On the Relation Between Approximation Fixpoint Theory and Justification Theory*. In Zhi-Hua Zhou, editor: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, ijcai.org, pp. 1973–1980, doi:10.24963/ijcai.2021/272. Available at `https://doi.org/10.24963/ijcai.2021/272`.

[13] Simon Marynissen, Bart Bogaerts, Marc Denecker & Jesse Heyninck (2022): *On Nested Justification Systems*. *Theory Pract. Log. Program.* 22. To appear (Accepted for ICLP 2022 special issue in TPLP).

[14] Simon Marynissen, Niko Passchyn, Bart Bogaerts & Marc Denecker (2018): *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. Available at `http://www4.uma.pt/nmr2018/NMR2018Proceedings.pdf`.

[15] Allen Van Gelder, Kenneth A. Ross & John S. Schlipf (1991): *The Well-Founded Semantics for General Logic Programs*. *J. ACM* 38(3), pp. 620–650, doi:10.1145/116825.116838. Available at `http://dx.doi.org/10.1145/116825.116838`.