Consistency in Justification Theory*

Simon Marynissen Niko Passchyn Bart Bogaerts Marc Denecker
KU Leuven, Departement of Computer Science, Leuven, Belgium

Abstract

Justification frameworks constitute a unifying theory to describe semantics of non-monotonic logics. They have applications, among others, in logic programming, abstract argumentation, and nested definitions. This framework is built on the notion of a justification. Intuitively, this is a graph that explains the truth values of certain facts. However this introduces a potential problem: the justification status of an atom and its negation can be conflicting. So, in a well-defined semantics we want that these statuses are opposite. This is captured in two axioms, which correspond to our main theorem: when a fact, positive or negative has a good justification, its negation should not have a good justification, and if a fact has no good justification, then its negation should have a good justification. We prove that this holds for completion, Kripke-Kleene, stable and well-founded semantics. The semantics of a justification framework is determined by what is called a branch evaluation. We introduce dual branch evaluations, which give rise to dual semantics, such as, co-well-founded and co-stable semantics. This duality is similar to the duality between induction and co-induction. Moreover, justifications in dual semantics are linked to justifications in the original one. Furthermore, we define a notion of splittable branch evaluation and show that under such an evaluation, justifications can be "glued" together, essentially resulting in a single justification for all true facts.

1 Introduction

In the field of Knowledge Representation (KR), novel logics and new semantics are constantly proposed. In order to keep an overview of the various logics, the semantics developed for a single logic and their relationships, unifying frameworks are developed (Denecker, Marek, and Truszczyński 2000; Denecker, Brewka, and Strass 2015). Such frameworks provide a uniform way to determine a broad class of semantics for all covered logics. Sometimes, this shines new light on the formal relationship between logics (Denecker, Marek, and Truszczyński 2011); sometimes, this gives rise to new semantics, such as for parametric versions of logic programs(Denecker, Brewka, and Strass 2015).

One such framework for non-monotonic logics is the justification theory from (Denecker, Brewka, and Strass 2015), which is built on the notion of a *justification*. Intuitively, this is a graph that explains the truth values of certain facts in a structure. Not any justification is considered "good". A good justification can be seen as an explanation for a fact. In order to determine the quality of a justification, justification theory makes use of *branch evaluations*. Different branch evaluations give rise to different semantics. Furthermore, so-called *nesting* of justification frames allows for a modular semantics and greatly reduces the efforts needed to introduce new language constructs (e.g., aggregates) into logics covered by justification theory. KR languages covered by justification theory include logic programs and answer set programs, abstract argumentation, inductive definitions, and nested definitions.

In addition, justifications are also used for implementation purposes. They are used to compute unfounded sets in modern ASP solvers (Gebser, Kaufmann, and Schaub 2012; De Cat et al. 2013), can be used to check for relevance of atoms in complete search algorithms (Jansen et al. 2016), and recent lazy grounding algorithms are built on top of them (De Cat et al. 2015; Bogaerts and Weinzierl 2018).

In all frameworks we know off, only truthness of atoms is explained. However, falsity of atoms, or equivalently truthness of negative literals, is not explained. This creates an asymmetry between atoms and negative literals. This asymmetry is resolved in justification theory by allowing justifications to also explain negative literals. However, this introduces a potential problem; the justification status of an atom and its negation can be conflicting. So, in a well-defined semantics we want that these statuses are opposite. This is captured in two axioms, which correspond to our main theorem: when a fact, positive or negative has a good justification, its negation should not have a good justification, and if a fact has no good justification, then its negation should have a good justification. This is obviously a fundamental problem, without which the framework fails. We investigate this matter in detail and positively answer this for completion, Kripke-Kleene, stable and well-founded semantics under the condition that the rules for a fact and its negation are related appropriately.

Besides this consistency of having a good justification, we introduce new concepts and prove results about these concepts. The first concept is splitting an interpretation into two sets of literals, which informally act as a lower and upper

*This paper is also submitted to the 13th Workshop on Logical and Semantic Frameworks with Applications (LSFA 2018).

bound on the interpretation. The two sets are each others dual in the sense that is defined in this text. This allows to reason on a level of sets instead of interpretations, which is easier from a mathematical point of view. Intuitively, however, one prefers to think in terms of interpretations. Most results are provided with an intuitive explanation in terms of interpretations.

Apart from duality on these sets, we also introduce a duality on branch evaluations, which amounts to a dual semantics in justification theory. One example is the duality between co-induction and induction, where the latter is tied with well-founded semantics. In addition, we also construct a co-stable semantics, which has the same relation with stable semantics as in the co-well-founded and well-founded case.

Moreover, we investigate when it is possible to "glue" good justifications together. To study this property, splittable and transitive branch evaluations are introduced, where the latter is a subclass of the former. Intuitively, these evaluations only depend on a start or tail of a branch. For splittable branch evaluations, we prove that "glueing" good justifications gives good justifications.

Denecker and Deschreye already established a result similar to our main theorem in (Denecker and De Schreye 1993) for tree-like justifications in a three-valued setting. Denecker et. al also use justifications for describing causal processes in an upcoming paper (Denecker, Bogaerts, and Vennekens 2018), and as such, our results are thus applicable for causal processes as well. The pasting of justifications introduces an operator between justifications. Similarly, in (Cabalar, Fandinno, and Fink 2014), an algebra of causal justifications is introduced.

The paper is structured as follows. In the preliminaries, we recall most of the definitions of justification theory, with small changes to ensure greater consistency and generality. In Section 3, we introduce complementary justification frames, which are based on complement closure as defined in (Denecker, Brewka, and Strass 2015). In Section 4, we first describe the lower and upper bounds on interpretations. Next, duality on these sets and dual branch evaluations are defined. Section 5 introduces splittable and transitive branch evaluations. In Section 6, we prove that justifications can be "glued" for splittable branch evaluations. We end in Section 7 by stating our main theorem. Due to page restrictions, proofs are omitted. A full version of this text, including most proofs, can be found at `https://bitbucket.org/simonmarynissen/consistencyinjustificationtheorynmr2018`. The results from Section 4 onwards were partially described in the master's thesis of the second author, see (Passchyn 2017), except for dual semantics and the proof of the main theorem in case of the well-founded semantics.

2 Preliminaries

In this section, we recall the basics of justification theory. Our presentation is based on the work of Denecker et al. (Denecker, Brewka, and Strass 2015). Small modifications to some existing definitions are made, to ensure greater consistency and generality; these changes have no major consequences and are clearly indicated.

In the rest of this paper, let $\mathcal{F}$ be a set, referred to as a *fact space*, such that $\mathcal{L} \coloneqq \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\} \subseteq \mathcal{F}$[1], where $\mathbf{t}$, $\mathbf{f}$, $\mathbf{u}$ and $\mathbf{i}$ have the respective meaning *true*, *false*, *unknown* and *inconsistent*. The elements of $\mathcal{F}$ are called *facts*. The set $\mathcal{L}$ is equal to the four-valued logic of Belnap (Belnap 1977) with truth order $\mathbf{f} \leq_t \mathbf{u}, \mathbf{i} \leq_t \mathbf{t}$ and information order $\mathbf{u} \leq_k \mathbf{f}, \mathbf{t} \leq_k \mathbf{i}$. 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 $\sim\mathbf{t} = \mathbf{f}$, $\sim\mathbf{u} = \mathbf{u}$, $\sim\mathbf{i} = \mathbf{i}$ and $\sim x \neq x$ for all $x \in \mathcal{F} \setminus \{\mathbf{u}, \mathbf{i}\}$. The assumption that $\sim\mathbf{u} = \mathbf{u}$ and $\sim\mathbf{i} = \mathbf{i}$ differs from (Denecker, Brewka, and Strass 2015), where a positive and negative $\mathbf{u}$ and $\mathbf{i}$ are used. For any fact x, $\sim x$ is called the *complement* of x. An example of a fact space $\mathcal{F}$ 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 containing elements of the form $\sim a$ for $a \in A$.

We distinguish two types of facts: *defined* and *open* facts[2]. 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: $\mathcal{L} \cap \mathcal{F}_d = \emptyset$;
- $R \subseteq \mathcal{F}_d \times 2^{\mathcal{F}}$;
- for each $x \in \mathcal{F}_d$, $(x, \emptyset) \notin R$ and there is an element $(x, A) \in R$.[3]

The set of *open* facts is denoted as $\mathcal{F}_o \coloneqq \mathcal{F} \setminus \mathcal{F}_d$. 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 $\mathcal{JF}$ is denoted as $\mathcal{JF}(x)$. Rules (x, A) are denoted as $x \leftarrow A$ and if $A = \{y_1, \ldots, y_n\}$, we often write $x \leftarrow y_1, \ldots, y_n$. Cases are allowed to be infinite; as such our framework covers also, for instance, non-ground logic programs where grounding can result in infinitely long bodies (as observed, e.g., by (Harrison, Lifschitz, and Yang 2013)).

A rule $x \leftarrow A$ expresses that if all facts of A are true, then x is true. Therefore, a case of x represents a sufficient condition for x to hold. Furthermore, the set of cases can be seen as a necessary condition for x to hold, that is, x only holds if there is a rule for x that can be applied. We illustrate the rest of notions and definitions with the the following running example.

[1] Denecker et al. (Denecker, Brewka, and Strass 2015) do not require that all logical facts are present in $\mathcal{F}$.

[2] In other literature, external, exogenous and parametric are also used to denote open facts. Defined facts are sometimes named internal or endogenous facts.

[3] Due to this property, justification frames here correspond to *proper* justification frames in (Denecker, Brewka, and Strass 2015). Any set of rules can be transformed into a proper justification frame; rules of the form $x \leftarrow \emptyset$ are replaced with $x \leftarrow \mathbf{t}$, and for $x \in \mathcal{F}_d$ that have no rules, we introduce $x \leftarrow \mathbf{f}$.

Example 2.2. In this example we build a justification frame to express the transitive closure of a graph. So let V be a set of nodes. Define $\mathcal{F}_o$ to be the set of elements $\mathrm{Edge}(v,w)$ and $\sim\mathrm{Edge}(v,w)$ with $v,w \in V$ and $\mathcal{F}_d$ to be the set of elements $\mathrm{Path}(v,w)$ and $\sim\mathrm{Path}(v,w)$ with $v,w \in V$. Define $\mathcal{F} = \mathcal{F}_d \cup \mathcal{F}_o$. The facts encoding the edges of a graph are in $\mathcal{F}_o$. This means that they can freely change and thus act as parameters, whereas the facts in $\mathcal{F}_d$ are constrained rules as seen below. In section 3 we construct the rules for $\sim\mathrm{Path}(v,w)$, but for now we only define the rules for $\mathrm{Path}(v,w)$:

- $\mathrm{Path}(v,w) \leftarrow \mathrm{Edge}(v,w)$;
- $\mathrm{Path}(v,w) \leftarrow \mathrm{Path}(v,x), \mathrm{Path}(x,w)$;

for all $v,w,x \in V$. This encodes that Path is the transitive closure of Edge. ▲

Definition 2.3. Two justification frames $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ and $\mathcal{JF}' = \langle \mathcal{F}, \mathcal{F}_d, R'\rangle$ are *equivalent* if for every rule $x \leftarrow A \in R$, there is a rule $x \leftarrow B \in R'$ such that $B \subseteq A$, and likewise for every rule $x \leftarrow B \in R'$, there is a rule $x \leftarrow A \in R$ such that $A \subseteq B$.

Equivalence is defined here in a different, but equivalent[4] way as in (Denecker, Brewka, and Strass 2015). A case $A \in \mathcal{JF}(x)$ so that there is a $B \in \mathcal{JF}(x)$ with $B \subseteq A$ is said to be *redundant*. Deleting a redundant rule results in an equivalent justification frame if the more general rule is kept. However, it is not always possible to remove all redundant rules without losing equivalence, see e.g. (Denecker, Brewka, and Strass 2015, Example 3).

Justifications and branch evaluations

Definition 2.4. A *justification* J in a justification frame $\langle \mathcal{F}, \mathcal{F}_d, R\rangle$ is a subset of R containing at most one rule for each $x \in \mathcal{F}_d$. If $x \leftarrow S \in J$, we denote $J(x) = S$.

This notation reveals that we can view a justification J as a partial function from $\mathcal{F}_d$ to $2^{\mathcal{F}}$ such that $x \leftarrow J(x) \in R$ if $x \in \mathrm{dom}(J)$, where $\mathrm{dom}(J)$ denotes the domain of J viewed as a total function. We use the two representations interchangeably.

A justification J gives a reason for facts x in its domain. This reason depends on a case of x. If this case $J(x)$ contains defined facts for which J does not give a reason, then our explanation is possibly incomplete. Therefore, we typically want that $J(x) \cap \mathcal{F}_d \subseteq \mathrm{dom}(J)$ for every $x \in \mathrm{dom}(J)$.

Definition 2.5. A justification is *locally complete*[5] if $J(x) \cap \mathcal{F}_d \subseteq \mathrm{dom}(J)$ for all $x \in \mathrm{dom}(J)$.

Given a justification J, we construct a directed graph G_J with vertices

$$\mathrm{dom}(J) \cup \{y \in \mathcal{F} \mid \exists x \in \mathrm{dom}(J) : y \in J(x)\}.$$

and edges

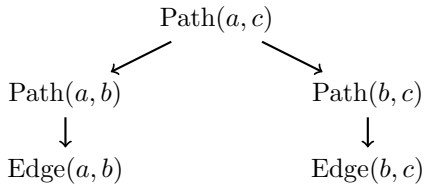
$$\{(x,y) \mid x \in \mathrm{dom}(J), y \in J(x)\}.$$

Since graphs are easily visualised, we obtain a natural and visual way to represent justifications.

Lemma 2.6. *A justification J is locally complete if and only if G_J has no defined leaves.*

Remark that the direction of the edges in G_J is opposite of the arrow direction in rule notation. We provide an example of how a justification could look like.

Example 2.7. Let $V = \{a,b,c\}$ in the setting of Example 2.2. Suppose that $\mathrm{Edge}(a,b)$ and $\mathrm{Edge}(b,c)$ hold. The following locally complete justification gives an explanation why $\mathrm{Path}(a,c)$ holds.



▲

Definition 2.8. Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ be a justification frame. A *$\mathcal{JF}$-branch* is either an infinite sequence in $\mathcal{F}_d$ or a finite non-empty sequence in $\mathcal{F}_d$ together with an element in $\mathcal{F}_o$. For a justification J in $\mathcal{JF}$, a *J-branch* starting in $x \in \mathcal{F}_d$ is a maximally long path (with at least one edge) in G_J starting in x.

For instance in Example 2.7, the path

$$\mathrm{Path}(a,c) \to \mathrm{Path}(a,b) \to \mathrm{Edge}(a,b)$$

is a finite branch of the given justification.

Remark 2.9. A finite $\mathcal{JF}$-branch can be represented as a finite sequence in $\mathcal{F}$ such that the last element is in $\mathcal{F}_o$ and all other elements in $\mathcal{F}_d$. As a consequence, a finite $\mathcal{JF}$-branch has at least two elements. Not all J-branches are $\mathcal{JF}$-branches since they can end in defined facts. However, if J is locally complete, any J-branch is also an $\mathcal{JF}$-branch by Lemma 2.6.

We denote a branch $\mathbf{b}$ as

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

and define $\sim\mathbf{b}$ as $\sim x_0 \to \sim x_1 \to \cdots$. A J-branch can be seen as part of a possible explanation of a fact. Given a rule $p \leftarrow p$, the explanation that p is true because p is true is generally not deemed acceptable. Therefore, a qualitative measure of branches is required.

Definition 2.10. A *branch evaluation* $\mathcal{B}$ is a mapping that maps any $\mathcal{JF}$-branch to an element in $\mathcal{F}$ for all justification frames $\mathcal{JF}$. A justification frame $\mathcal{JF}$ together with a branch evaluation $\mathcal{B}$ form a *justification system* $\mathcal{JS}$, which is presented as a quadruple $\langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}\rangle$.

We start with some particular branch evaluations.

Example 2.11. The *supported* (or completion (Clark 1978)) branch evaluation $\mathcal{B}_{\mathrm{sp}}$ maps $x_0 \to x_1 \to \cdots$ to x_1. The *Kripke-Kleene* branch evaluation $\mathcal{B}_{\mathrm{KK}}$ maps finite branches to their last node and infinite branches to $\mathbf{u}$. ▲

[4] Equivalence only holds if the derivation operator in (Denecker, Brewka, and Strass 2015) is extended to all subsets of $\mathcal{F}$ in the obvious way.

[5] There is also the notion of a complete justification: $\mathcal{F}_d = \mathrm{dom}(J)$. However, such a justification has a rule for any defined fact and thus tries to explain every fact, even complementary facts.

The names of these branch evaluations are not arbitrary and the correspondence with the equally named semantics for logic programs is explained in (Denecker, Brewka, and Strass 2015). Later, we also define well-founded and stable branch evaluations. These examples show that a branch evaluation induces a semantics. This is manifested by fixed points of the extended support operator defined below, which is illustrated in (Denecker, Brewka, and Strass 2015).

Definition 2.12. Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a justification system, I a subset[6] of $\mathcal{F}$ and J a locally complete justification in $\mathcal{JS}$. We say $x \in \mathcal{F}_d$ is *supported* by J in I under $\mathcal{B}$ if $x \in \mathrm{dom}(J)$ and all J-branches starting in x are mapped into I under $\mathcal{B}$. The fact x is *supported* by $\mathcal{JS}$ in I if there exists a locally complete[7] justification J in $\mathcal{JS}$ that supports x in I under $\mathcal{B}$.

This definition is best clarified with an example.

Example 2.13. Let the setting be the same as in Example 2.7 and set $I = \{\mathrm{Edge}(a,b), \mathrm{Edge}(b,c)\}$. The fact $\mathrm{Path}(a,c)$ is supported in I under the Kripke-Kleene branch evaluation, a justification supporting $\mathrm{Path}(a,c)$ is already given in Example 2.7. ▲

Associated with a justification system, we get the *support operator* $\mathcal{S}_{\mathcal{JS}} : 2^{\mathcal{F}} \to 2^{\mathcal{F}_d}$ that maps $I \subseteq \mathcal{F}$ to

$$\mathcal{S}_{\mathcal{JS}}(I) \coloneqq \{x \in \mathcal{F}_d \mid x \text{ is supported by } \mathcal{JS} \text{ in } I\}.$$

If $\mathcal{JS}$ consists of a justification frame $\mathcal{JF}$ and a branch evaluation $\mathcal{B}$, this operator is also denoted as $\mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}$. The support operator can be extended to return subsets of $\mathcal{F}$ as follows:

$$\hat{\mathcal{S}}_{\mathcal{JS}} : 2^{\mathcal{F}} \to 2^{\mathcal{F}} : I \mapsto \mathcal{S}_{\mathcal{JS}}(I) \cup (I \cap \mathcal{F}_o).$$

In (Denecker, Brewka, and Strass 2015), various semantics are linked to fixed points of this *extended support operator*.

We also define what it means for a defined fact to be supported by a non locally complete justification.

Definition 2.14. A fact $x \in \mathcal{F}_d$ is *supported* by a (not necessarily locally complete) justification J in I under $\mathcal{B}$ if every locally complete extension of J supports x in I under $\mathcal{B}$.

This definition of support is compatible with the previous one since if J is a locally complete justification and K a locally complete extension of J, then any K-branch starting in $x \in \mathrm{dom}(J)$ is also a J-branch; hence K supports all facts that J supports. This alternative definition is useful in proofs, but also allows to give justifications that do not contain unnecessary information. For instance, under completion semantics, we do not need locally complete justifications, but just a single rule.

Support is invariant under equivalence of the underlying justification frame.

Proposition 2.15. *Let $\mathcal{JF}$ and $\mathcal{JF}'$ be two equivalent justification frames. Then $\mathcal{S}^{\mathcal{B}}_{\mathcal{JF}} = \mathcal{S}^{\mathcal{B}}_{\mathcal{JF}'}$ for any branch evaluation $\mathcal{B}$.*

[6] In (Denecker, Brewka, and Strass 2015), this is only defined for subsets containing $\mathbf{t}$ and $\mathbf{i}$ and not $\mathbf{f}$ and $\mathbf{u}$. The reason to allow more sets is explained later in subsection 4.

[7] In (Denecker, Brewka, and Strass 2015), a complete justification is required, that is, the domain is equal to $\mathcal{F}_d$. This, however, does not change the definition of support.

3 Complementary justification frames

In many applications, only one of x or $\sim x$ has explicit rules. For instance in logic programming (van Emden and Kowalski 1976; Marek and Truszczyński 1999), only rules for atoms are given, the rules for negative literals are left implicit. On the other hand, in Dungs argumentation frameworks (Dung 1995), only an attack relation between arguments is given, which can be seen as explicit rules for falsity of arguments, e.g., if a attacks b, then this is expressed by the rule $\sim b \leftarrow a$. Even in Example 2.2 we refrained from formulating when $\sim\mathrm{Path}(v,w)$ holds.

Under the assumption that x and $\sim x$ are complementary, rules for $\sim x$ can be constructed by using rules for x. As mentioned before, a case S of x can be seen as a sufficient condition for x to hold. Similarly, x only holds if a case of x holds, thus $\sim x$ holds if every case of x has a a fact that does not hold.

Definition 3.1. A *selection function* of $x \in \mathcal{F}_d$ in a justification frame $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \rangle$ is a function $\mathcal{S} : \mathcal{JF}(x) \to \mathcal{F}$ such that $\mathcal{S}(A) \in A$ for all $A \in \mathcal{JF}(x)$.

A selection function for x selects an element of each rule of x. In general, selection functions exists assuming the axiom of choice. Given a selection function $\mathcal{S}$ of x, the set $\sim\mathrm{Im}(\mathcal{S})$ is a potential case of $\sim x$, where $\mathrm{Im}(\mathcal{S})$ is the image of $\mathcal{S}$. This motivates the following.

Definition 3.2. Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \rangle$ be a justification frame. Define $R^* \subseteq \mathcal{F}_d \times 2^{\mathcal{F}}$ as the set of elements of the form

$$(\sim x, \sim\mathrm{Im}(\mathcal{S}))$$

for $x \in \mathcal{F}_d$ and $\mathcal{S}$ a selection function of x. The *complement* of $\mathcal{JF}$ is defined as $\mathrm{C}(\mathcal{JF}) \coloneqq \langle \mathcal{F}, \mathcal{F}_d, R^* \rangle$. The *complementation* of $\mathcal{JF}$ is defined as $\mathrm{CC}(\mathcal{JF}) \coloneqq \langle \mathcal{F}, \mathcal{F}_d, R \cup R^* \rangle$. A justification frame $\mathcal{JF}$ is *complementary* if it is equivalent with $\mathrm{C}(\mathcal{JF})$.

Remark 3.3. Denecker et al. (Denecker, Brewka, and Strass 2015) only defined complementation, which they called complement closure, which is a misleading name since complementation is not a closure, i.e. not idempotent. The essential properties of complementation are captured by complementary justification frames.

Complementation can also be applied to any set of rules. Most examples will start with a set or rules that does not form a justification frame.

Example 3.4. Take $\mathcal{F}_d = \{a, b, c, \sim a, \sim b, \sim c\}$, $\mathcal{F}_o = \mathcal{L}$ and $R = \{a \leftarrow b; a \leftarrow c; b \leftarrow \mathbf{t}; c \leftarrow \mathbf{t}\}$, then the complementation of R gives a complementary justification frame $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \cup R^* \rangle$, which adds the rules $\{\sim a \leftarrow \sim b, \sim c; \sim b \leftarrow \mathbf{f}; \sim c \leftarrow \mathbf{f}\}$. For instance the rule $\sim a \leftarrow \sim b, \sim c$ is in R since $a \leftarrow b$ and $a \leftarrow c$ are in R. ▲

Let us also check what complementation means in our running example.

Example 3.5. In Example 2.2, we did not state the rules for $\sim\mathrm{Path}(v,w)$. With complementation we get

$$\sim\mathrm{Path}(v,w) \leftarrow \{\sim\mathrm{Edge}(v,w)\} \cup S$$

with S a subset of $\mathcal{F}$ such that for every $x \in V$, at least $\sim\mathrm{Path}(v,x)$ or $\sim\mathrm{Path}(x,w)$ is in S. Intuitively it says that v and w are not connected if there is no edge between v and w and for all $x \in V$, there is no path from v to x or no path from x to w. ▲

We obtain the following characterisation for complementary frames.

Proposition 3.6. *Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ be a justification frame. Then $\mathcal{JF}$ is complementary if and only if for every $x \in \mathcal{F}_d$ the following hold*

1. *for every selection function $\mathcal{S}$ of x in $\mathcal{JF}$, there exists an $A \in \mathcal{JF}(\sim x)$ such that $A \subseteq \sim\mathrm{Im}(\mathcal{S})$;*
2. *for every $A \in \mathcal{JF}(x)$, there exists a selection function $\mathcal{S}$ of $\sim x$ in $\mathcal{JF}$ such that $\sim\mathrm{Im}(\mathcal{S}) \subseteq A$.*

Proof. Follows directly from writing out what equivalence means in this context. □

With complementation we can construct complementary justification frames.

Proposition 3.7. *Let $\mathcal{F}$ be a fact space and let $\mathcal{F}_d$ be a subset of $\mathcal{F}$ closed under $\sim$ that does not contain logical facts. Let $\{\mathcal{F}_1, \mathcal{F}_2\}$ be a partition of $\mathcal{F}_d$ such that $\sim\mathcal{F}_1 = \mathcal{F}_2$. Let R be a subset of $\mathcal{F}_1 \times (2^{\mathcal{F}} \setminus \emptyset)$ such that for each $x \in \mathcal{F}_1$, there is an $(x, A) \in R$. Then $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \cup R^*\rangle$ is complementary.*

In general, complementation of a justification frame is not always complementary as illustrated by the following example.

Example 3.8. Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ with $\mathcal{F}_d = \{a, \sim a\}$, $\mathcal{F}_o = \mathcal{L}$ and $R = \{a \leftarrow \mathbf{t}; \sim a \leftarrow \mathbf{t}\}$. Then the complementation of $\mathcal{JF}$ has the following rules $\{a \leftarrow \mathbf{t}; \sim a \leftarrow \mathbf{t}; a \leftarrow \mathbf{f}; \sim a \leftarrow \mathbf{f}\}$, while the complement of the complementation of $\mathcal{JF}$ has the rules $\{a \leftarrow \mathbf{t}, \mathbf{f}; \sim a \leftarrow \mathbf{t}, \mathbf{f}\}$. These two justification frames are not equivalent; hence the complementation of $\mathcal{JF}$ is not complementary. ▲

The following lemma illustrates that cases of a fact and its negation are intrinsically related in a complementary justification frame.

Lemma 3.9. *If a justification frame is complementary, then for every rule $x \leftarrow A$ and rule $\sim x \leftarrow B$, $A \cap \sim B \neq \emptyset$.*

Lemma 3.9 expands to justifications.

Lemma 3.10. *Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ be a complementary justification frame and $x \in \mathcal{F}_d$. If J and K are locally complete justifications with $x \in \mathrm{dom}(J)$ and $\sim x \in \mathrm{dom}(K)$, then there exists a J-branch $\mathbf{b}$ starting in x such that $\sim\mathbf{b}$ is a K-branch starting in $\sim x$.*

Even though, we have found branches that are each others negation, in general we cannot relate the evaluation of these branches. The following definition gives an obvious relation.

Definition 3.11. A branch evaluation $\mathcal{B}$ is *consistent* if for all justification frames $\mathcal{JF}$ and all $\mathcal{JF}$-branches $\mathbf{b}$ it holds that $\mathcal{B}(\sim\mathbf{b}) = \sim\mathcal{B}(\mathbf{b})$.

If a justification frame is complementary it means that rules for a fact and its negation are compatible. One should expect that in a such a frame, the justification status should also be compatible, that is, a fact is supported in I if and only if the negation is not supported in I. However, this is not always the case, as illustrated in the following example.

Example 3.12. Take $\mathcal{F}_d = \{a, \sim a, b, \sim b, c, \sim c\}$ and $\mathcal{F}_o = \mathcal{L}$, $\mathcal{F}_+ = \{a, b, c\}$, and $\mathcal{F}_- = \{\sim a, \sim b, \sim c\}$. The set of rules is the complementation of

$$\{a \leftarrow b; a \leftarrow c; b \leftarrow a; c \leftarrow a\}\,.$$

Define the branch evaluation $\mathcal{B}$ as follows for infinite branches:

- $\mathcal{B}(x_0 \to x_1 \to \cdots) = \mathbf{f}$ if $\exists i_0 \in \mathbb{N} : \forall i, j > i_0 : x_i \in \mathcal{F}_+$ and if $x_i = x_j$, then $x_{i+1} = x_{j+1}$;
- $\mathcal{B}(x_0 \to x_1 \to \cdots) = \mathbf{t}$ if $\exists i_0 \in \mathbb{N} : \forall i, j > i_0 : x_i \in \mathcal{F}_-$ and if $x_i = x_j$, then $x_{i+1} = x_{j+1}$;
- $\mathcal{B}(x_0 \to x_1 \to \cdots) = \mathbf{u}$ otherwise.

Finite branches are mapped to their last element. The only connected locally complete justifications with a or $\sim a$ in their domain are given below:

Therefore, it is easy to see that a is not supported by $\mathcal{JS}$ in any subset of $\mathcal{F}$ not containing $\mathbf{f}$, but also $\sim a$ is not supported by $\mathcal{JS}$ in any subset of $\mathcal{F}$ not containing both $\mathbf{t}$ and $\mathbf{u}$. This means that the semantics of the branch evaluation $\mathcal{B}$ is defective. The goal of the rest of the paper is to prove that this situation cannot occur for supported, Kripke-Kleene, stable and well-founded branch evaluations. ▲

4 Duality

Duality results are ubiquitous in mathematics and computer science. Duality results of points and hyperplanes in projective geometry are famous examples. In logic, the duality between conjunctive normal form (CNF) and disjunctive normal form (DNF) is another. This duality can be explained in terms of justification frames. Let ϕ be a CNF formula, i.e. it is a conjunction of clauses, of which each clause is a disjunction of literals. Let $c_1, \ldots, c_n$ be the different conjuncts of ϕ. For each conjunct c_i, let $d_{i,1}, \ldots, d_{i,m_i}$ be its disjuncts. We can construct the following set R of rules containing

$$\phi \leftarrow c_1, \ldots, c_n \quad \text{and} \quad c_i \leftarrow d_{i,j}$$

for all $1 \leq i \leq n$ and $1 \leq j \leq m_i$. The set R corresponds to the CNF formula ϕ. The dual of ϕ is defined as ϕ after swapping $\wedge$ with $\vee$ and changing literals to their negation. The complementation of the above set R forms a justification frame if taking $\mathcal{F}_d = \{\phi, \sim\phi, c_1, \sim c_1, \ldots, c_n, \sim c_n\}$ and $\mathcal{F}_o = \mathcal{L} \cup \{d_{i,j}, \sim d_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\}$. The rules for $\sim\phi$ and $\sim c_i$ correspond with the dual of the CNF formula ϕ. The duality result between CNF and DNF is as follows:

- ϕ is satisfiable if the dual of ϕ is not valid.

- ϕ is valid if the dual of ϕ is not satisfiable.

Let I be a subset of $\mathcal{F}$ such that exactly one of $d_{i,j}$ and $\sim d_{i,j}$ is in I for all i and j. Call I a $*$-set. It is not hard to see that ϕ is satisfiable if and only if ϕ is supported in some $*$-set I under $\mathcal{B}$ with $\mathcal{B}$ any branch evaluation that maps finite branches to their last element. For simplicity, suppose $\mathcal{B} = \mathcal{B}_{\mathrm{KK}}$. Similarly, the dual of ϕ is satisfiable if and only if $\sim\phi$ is supported in some $*$-set I under $\mathcal{B}$. This means that we have that

- ϕ is supported in some $*$-set I if and only if $\sim\phi$ is not supported in some $*$-set I.
- ϕ is supported in all $*$-sets I if and only if $\sim\phi$ is not supported in all $*$-sets I.

In what follows, we investigate how this result generalizes, i.e., we investigate the following question. If a fact x is supported in I, will $\sim x$ be supported in some I'? And conversely, if a fact x is not supported in I, will $\sim x$ be supported in some I'? We investigate this when I' is the dual of I, which we define below. The duality result of CNF and DNF formulas is also a consequence of our main theorem.

Interpretable sets

In this section we show how "subsets of $\mathcal{F}$" relate to interpretations as often defined in various branches of logic. In particular, given a propositional vocabulary Σ, a four-valued interpretation is often defined as a mapping $\Sigma \to \mathcal{L}$. Taking $\mathcal{F}$ the set of literals over Σ, extended with the logical facts, this induces a map $\mathcal{I} : \mathcal{F} \to \mathcal{L}$ such that $\mathcal{I}(\sim x) = \sim\mathcal{I}(x)$ for all $x \in \mathcal{F}$ and $\mathcal{I}(\ell) = \ell$ for all $\ell \in \mathcal{L}$. We call such a map an interpretation of $\mathcal{F}$.

As mentioned before, our set of logical facts corresponds to Belnap's four-valued logic (Belnap 1977). Each value can also be represented as a closed interval (in the truth order, represented as a tuple) of two-valued truth values: $\mathbf{t}$ corresponds to $(\mathbf{t},\mathbf{t})$, $\mathbf{f}$ to $(\mathbf{f},\mathbf{f})$, $\mathbf{u}$ to $(\mathbf{f},\mathbf{t})$, and $\mathbf{i}$ to $(\mathbf{t},\mathbf{f})$.

For each interpretation $\mathcal{I}$, we can associate two sets of facts: a *lower interpretable subset* I_ℓ and an *upper interpretable subset* I_u. The set I_ℓ consists of the true and inconsistent facts and I_u consists of the true and unknown facts. We can construct each type of fact. This is summarised below.

- The true facts are $I_\ell \cap I_u$.
- The false facts are $\mathcal{F} \setminus (I_\ell \cup I_u)$.
- The unknown facts are $I_u \setminus I_\ell$.
- The inconsistent facts are $I_\ell \setminus I_u$.

Remark 4.1. The terms lower and upper refer to which coordinate should be $\mathbf{t}$. For instance, the lower interpretable set consists of the true and inconsistent facts: the only logical facts for which the first coordinate is $\mathbf{t}$ in the above mentioned tuple notation are $\mathbf{t}$ and $\mathbf{i}$.

Now take an arbitrary subset I of $\mathcal{F}$. Let L be the set of logical facts in I. If $L = \{\mathbf{t},\mathbf{i}\}$, then I corresponds to the lower interpretable subset of a unique interpretation $\mathcal{I}_I$. Similar, if $L = \{\mathbf{t},\mathbf{u}\}$, then I corresponds to the upper interpretable subset of a unique interpretation $\mathcal{I}_I$. These are the only possibilities for interpretable subsets.

The set of interpretable[8] subsets of $\mathcal{F}$ is denoted as $\mathbb{I}_\mathcal{F}$ (or by $\mathbb{I}$ if $\mathcal{F}$ is clear from the context).

Definition 4.2. An interpretable subset I of $\mathcal{F}$ is *consistent* if $\mathcal{I}_I(x) = \mathbf{i}$ if and only if $x = \mathbf{i}$. An interpretable subset I of $\mathcal{F}$ is *co-consistent* if $\mathcal{I}_I(x) = \mathbf{u}$ if and only if $x = \mathbf{u}$. An interpretable subset I of $\mathcal{F}$ is *exact* if it is consistent and co-consistent.

For a given interpretation, the lower and upper interpretable subsets are related to each other by the following duality operator.

Definition 4.3. We define the *duality operator* on subsets of $\mathcal{F}$ as follows:

$$\mathrm{d}_\mathcal{F} : 2^\mathcal{F} \to 2^\mathcal{F} : I \mapsto \mathrm{d}_\mathcal{F}(I) \coloneqq \sim(\mathcal{F} \setminus I) = \mathcal{F} \setminus \sim I.$$

When $\mathcal{F}$ is clear from the context, we just write d for $\mathrm{d}_\mathcal{F}$. For an interpretation $\mathcal{I}$, we have that $\mathrm{d}(I_\ell) = I_u$ and $\mathrm{d}(I_u) = I_\ell$. In general the following holds.

Proposition 4.4. *1. The duality operator is involutive, that is* $\mathrm{d}(\mathrm{d}(I)) = I$ *for all* $I \subseteq \mathcal{F}$.

2. *The duality operator maps lower interpretable subsets to upper interpretable subsets and vice versa.*
3. *For I interpretable, it holds that $\mathcal{I}_{\mathrm{d}(I)} = \mathcal{I}_I$.*
4. *The duality operator preserves consistency and co-consistency of interpretable subsets.*
5. *The dual of an exact interpretable subset I differs from I only on the logical facts.*

Item (3) shows that the dual of an interpretable subset provides a different view on the same object (the same underlying interpretation), as is often the case with dualities in mathematics.

Remark 4.5. We effectively split an interpretation in a lower and upper interpretable subsets. Unfortunately, these concepts are less intuitive than interpretations. On the positive side, they simplify the arguments for proofs and enable us to state results that are not obvious in terms of interpretations. In order to relate to intuitions, we will rephrase most results on interpretable sets below in terms of interpretations.

Let $\mathcal{JS}$ be a justification system and $\mathcal{I}$ and interpretation of $\mathcal{F}$.

Definition 4.6. Let $\mathcal{JS}$ be a justification system, $x \in \mathcal{F}$ a fact, and $\mathcal{I}$ an interpretation of $\mathcal{F}$. If x is defined, the *supported value* of x in $\mathcal{I}$ by $\mathcal{JS}$ is defined as:

$$\mathrm{SV}_\mathcal{JS}(\mathcal{I}, x) \coloneqq \bigvee_{J \in \mathcal{J}_\mathcal{JS}(x)} \bigwedge_{\mathbf{b} \in B_J(x)} \mathcal{I}(\mathcal{B}(\mathbf{b})),$$

where $\mathcal{J}_\mathcal{JS}(x)$ is the set of locally complete justifications J in $\mathcal{JS}$ with $x \in \mathrm{dom}(J)$ and $B_J(x)$ is the set of J-branches starting in x. The meet and join are with respect to the truth order.

If x is open, the supported value of x in $\mathcal{I}$ by $\mathcal{JS}$ is defined as $\mathrm{SV}_\mathcal{JS}(\mathcal{I}, x) \coloneqq \mathcal{I}(x)$.

[8] In (Denecker, Brewka, and Strass 2015), Denecker et al. only considered lower interpretable subsets as input for the support operator. They called those sets interpretations. We chose for the terminology 'interpretable' to emphasise the difference with what is usually called an interpretation in logic.

This value can be seen as the value of the best justification for x. For a given justification J, $\bigwedge_{\mathbf{b}\in B_J(x)} \mathcal{I}(\mathcal{B}(\mathbf{b}))$ can intuitively be seen as the worst value of its branches.

It is not too difficult to see that the supported value is the same as the following tuple of two-valued truth values

$$(x \in \hat{\mathcal{S}}_{\mathcal{JS}}(I_\ell), x \in \hat{\mathcal{S}}_{\mathcal{JS}}(I_u)),$$

where the expression $x \in A$ is considered $\mathbf{t}$ if x is in A and $\mathbf{f}$ if not. As stated in the introduction, we want that the supported value commutes with negation, i.e. $\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, x) = \sim \mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \sim x)$. To prove this for all interpretations, this amounts to proving that x is supported in I if and only if $\sim x$ is not supported in $\mathrm{d}(I)$ for any interpretable set I. In algebraical terms, this is exactly

$$\mathrm{d}(\hat{\mathcal{S}}_{\mathcal{JS}}(I)) = \hat{\mathcal{S}}_{\mathcal{JS}}(\mathrm{d}(I))$$

for all interpretable subsets.

Remark 4.7. This rewording directly shows that our main theorem implies the duality between CNF and DNF formulas as mentioned in the introduction of this section.

If the supported value commutes with negation, then the supported value is the value of the interpretation corresponding to $\hat{\mathcal{S}}_{\mathcal{JS}}(I)$, which is well defined since $\hat{\mathcal{S}}_{\mathcal{JS}}(I)$ is interpretable since I is interpretable. In what follows we study this property and prove it for completion, Kripke-Kleene, stable and well-founded semantics. Moreover, we will prove it for co-stable and co-well-founded semantics, which we define in the next two subsections.

Signed justification frames

In many logical frameworks, there is an asymmetry between positive and negative literals, while in justification theory, facts and their negation are treated completely symmetrically. In this section, we show how a similar distinction can be introduced by means of a sign function. Consider for instance stable semantics of logic programs (Gelfond and Lifschitz 1988). There, the default value for atoms is false; as such, the default value for its negation is true. Thus, the set of defined literals is divided into two parts, those that are default and those that are not.[9] This idea is captured in signed justification frames.

Definition 4.8. Let $\mathcal{JF}$ be a justification frame. A *sign function* on $\mathcal{JF}$ is a map

$$\mathrm{sgn} : \mathcal{F}_d \to \{-, +\}$$

such that $\mathrm{sgn}(x) \neq \mathrm{sgn}(\sim x)$ for all $x \in \mathcal{F}_d$. We denote $\mathcal{F}_- := \mathrm{sgn}^{-1}(\{-\})$ and $\mathcal{F}_+ := \mathrm{sgn}^{-1}(\{+\})$. Remark that $\{\mathcal{F}_-, \mathcal{F}_+\}$ forms a partition of $\mathcal{F}_d$.

From now on, we fix a sign function on $\mathcal{JF}$. This structure is always present in a justification frame defined as in (Denecker, Brewka, and Strass 2015) and there it is a partition of whole $\mathcal{F}$. However, it raises the question which logical facts are positive and which are negative. To solve this, Denecker et al. introduced a positive and negative $\mathbf{u}$ and $\mathbf{i}$, which is an ad hoc solution and deviates from Belnap's four-valued logic (Belnap 1977). We abstain from deciding which logical fact is positive by only introducing it for defined facts.

The sign function is used to define new branch evaluations.

Definition 4.9. The *well-founded* (Van Gelder, Ross, and Schlipf 1991) branch evaluation $\mathcal{B}_{\mathrm{wf}}$ maps finite branches to their last element. Under $\mathcal{B}_{\mathrm{wf}}$, infinite branches are mapped 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* (Gelfond and Lifschitz 1988) (answer set) branch evaluation[10] $\mathcal{B}_{\mathrm{st}}$ 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 to $\mathbf{t}$ if x_0 is negative and $\mathbf{f}$ if x_0 is positive. The only exception is that a finite branch $x_0 \to \cdots \to x_n$, with $x_0, \ldots, x_{n-1}$ having the same sign, is mapped to x_n.

The correspondence with the corresponding logic programming semantics is given in the following theorem from (Denecker, Brewka, and Strass 2015, Theorem 1).

Theorem 4.10 (Denecker et al.)**.** *To a propositional logic program* Π *we can associate a complementary justification frame* $\mathcal{JF}_\Pi$ *such that*

- *An exact lower interpretable subset* I *is an exact fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{sp}}}_{\mathcal{JF}_\Pi}$ *iff* I *is a supported model of* Π.
- *A lower interpretable subset* I *is a fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{KK}}}_{\mathcal{JF}_\Pi}$ *iff* I *is the Kripke-Kleene model of* Π.
- *A lower interpretable subset* I *is an exact fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{st}}}_{\mathcal{JF}_\Pi}$ *iff* I *is a stable model of* Π.
- *A lower interpretable subset* I *is a fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{wf}}}_{\mathcal{JF}_\Pi}$ *iff* I *is the well-founded model of* Π.

Dual branch evaluations

Well-founded semantics accurately captures the semantics of inductive definitions (Denecker and Ternovska 2004; Denecker and Vennekens 2007; Denecker and Ternovska 2008; Denecker and Vennekens 2014). On the other hand, there are co-inductive definitions (Gupta et al. 2007). We define a semantics for co-induction by introducing a co-well-founded branch evaluation[11].

Definition 4.11. Let $\mathcal{JF}$ be a justification frame. Then the *co-well-founded* branch evaluation $\mathcal{B}_{\mathrm{cwf}}$ maps finite branches to their last element, branches with a positive tail to $\mathbf{t}$, branches with a negative tail to $\mathbf{f}$ and all other branches to $\mathbf{u}$.

After close inspection, this is exactly $\mathcal{B}_{\mathrm{wf}}$, but with an inverted sign function. This motivates the following. If $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ is a justification frame with sign function $\mathrm{sgn}_{\mathcal{JF}}$, then we denote $\overline{\mathcal{JF}}$ to be the justification frame $\langle \mathcal{F}, \mathcal{F}_d, R\rangle$ with sign function $\mathrm{sgn}_{\overline{\mathcal{JF}}} := -\mathrm{sgn}_{\mathcal{JF}}$. Each $\mathcal{JF}$-branch $\mathbf{b}$

[9] In some contexts, non-default literals are called *deviant* (Denecker, Bogaerts, and Vennekens 2018).

[10] This is different on finite branches than the stable branch evaluation introduced in (Denecker, Brewka, and Strass 2015); however, fixed points of the extended support operators are equal.

[11] Denecker et. al first described the co-well-founded branch evaluation in (Denecker, Brewka, and Strass 2015).

is a $\overline{\mathcal{JF}}$-branch. To make explicit the justification frame $\mathcal{JF}$ we regard a branch $\mathbf{b}$ in, we denote $\mathbf{b}$ as $\mathbf{b}_{\mathcal{JF}}$.

Definition 4.12. Let $\mathcal{B}$ be a branch evaluation. The *dual branch evaluation* $\overline{\mathcal{B}}$ of $\mathcal{B}$ is defined as follows:

$$\overline{\mathcal{B}}(\mathbf{b}_{\mathcal{JF}}) = \mathcal{B}(\mathbf{b}_{\overline{\mathcal{JF}}})$$

for any justification frame $\mathcal{JF}$ and $\mathcal{JF}$-branch $\mathbf{b}$.

We chose to call it dual semantics as induction and co-induction are dual concepts.

Example 4.13. The well-founded and co-well-founded branch evaluation are each others dual. ▲

Branch evaluations, such as $\mathcal{B}_{\mathrm{sp}}$ and $\mathcal{B}_{\mathrm{KK}}$, that do not depend on a sign function are self-dual. Apart from the co-well-founded branch evaluation, we constructed a new semantics: the *co-stable* branch evaluation $\mathcal{B}_{\mathrm{cst}} \coloneqq \overline{\mathcal{B}_{\mathrm{st}}}$. It differs with $\mathcal{B}_{\mathrm{st}}$ only on infinite branches having everywhere the same sign. In this case, it is the negation of $\mathcal{B}_{\mathrm{st}}$. Where stable semantics gives a default value of false to atoms, co-stable semantics gives a default value of true to atoms. Co-stable semantics could be useful in the context of stream reasoning (Beck, Eiter, and Folie 2017).

Justifications in dual semantics are related with justifications in the original semantics as illustrated in the following lemma.

Lemma 4.14. *Let $\mathcal{JS}$ be a justification system $\langle \mathcal{JF}, \mathcal{B} \rangle$ and let $\mathcal{JS}'$ be the justification system $\langle \overline{\mathcal{JF}}, \overline{\mathcal{B}} \rangle$. Then $\mathcal{S}_{\mathcal{JS}}(I) = \mathcal{S}_{\mathcal{JS}'}(I)$ for all subsets I of $\mathcal{F}$.*

While the proof of this lemma is (almost) trivial, it has a consequence that relates commutation of negation with the supported value between dual branch evaluations.

Corollary 4.15. *For a subset I of $\mathcal{F}$, the following are equivalent.*

1. *Under $\mathcal{B}$, x is supported in I if and only if $\sim x$ is not supported in $\mathrm{d}(I)$. ($\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(\mathrm{d}(I)) = \mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I))$)*
2. *Under $\overline{\mathcal{B}}$, x is supported in I if and only if $\sim x$ is not supported in $\mathrm{d}(I)$. ($\hat{\mathcal{S}}^{\overline{\mathcal{B}}}_{\mathcal{JF}}(\mathrm{d}(I)) = \mathrm{d}(\hat{\mathcal{S}}^{\overline{\mathcal{B}}}_{\mathcal{JF}}(I))$)*

Thus if we prove that the supported value commutes with negation for stable and well-founded semantics, we automatically have that the supported value of co-stable and co-well-founded semantics commutes with negation.

5 Branch evaluation classes

In this section, we study properties of various branch evaluations, in particular $\mathcal{B}_{\mathrm{sp}}$, $\mathcal{B}_{\mathrm{KK}}$, $\mathcal{B}_{\mathrm{wf}}$, $\mathcal{B}_{\mathrm{st}}$, and their duals. To do this systematically, we introduce novel classes of branch evaluations and study properties of them. In particular, we study so-called *splittable* branch evaluations. Such a branch evaluation can evaluate any branch either by only taking its start (first n elements) or its tail into account. Splittable branch evaluations allow us to "glue" justifications together and as such construct a single justification that supports $\mathcal{S}_{\mathcal{JS}}(I)$ in I.

For a finite branch $\mathbf{b} : x_0 \to x_1 \to \cdots \to x_n$, denote $\ell(\mathbf{b}) = n$. 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 *decided* under $\mathcal{B}$ at $0 < n < \ell(\mathbf{b}) + 1$ if for every $\mathcal{JF}$-branch $\mathbf{b}'$ identical to $\mathbf{b}$ up to n, we have $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{b}')$. In this case, we call the $\mathcal{JF}$-path $x_0 \to \cdots \to x_n$ *decided* under $\mathcal{B}$. This means to determine the value of $\mathbf{b}$ under $\mathcal{B}$, we only need the $n+1$ first elements, so all relevant information is located at the beginning of the branch. This allows us to extend $\mathcal{B}$ to paths that are decided. On the other hand, we define $\mathbf{b}$ to be *transitive* 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 transitive if all information needed to evaluate it is located in the tail after n.

Definition 5.1. A branch $\mathbf{b}$ is called *splittable* under $\mathcal{B}$ at $0 \leq n < \ell(\mathbf{b}) + 1$ if it is either decided or transitive under $\mathcal{B}$ at n.

Intuitively, if a branch is splittable at n, then the information to evaluate the branch is either in tail or start, but not in both.

Remark 5.2. If $\mathcal{B}$ is clear from context, 'under $\mathcal{B}$' is left out.

Lemma 5.3. *If a branch $\mathbf{b}$ is decided at n, then it is also decided at m for $n \leq m < \ell(\mathbf{b}) + 1$.*

A branch $\mathbf{b}$ is called *first decided* at $0 < i < \ell(\mathbf{b}) + 1$ if $\mathbf{b}$ is decided at $i = 1$ or $\mathbf{b}$ is decided at $i > 1$ and $\mathbf{b}$ is not decided at $i - 1$.

Definition 5.4. We say that a branch $\mathbf{b}$ is *totally decided* if it is decided at i for every $0 < i < \ell(\mathbf{b}) + 1$. Similarly, $\mathbf{b}$ is *totally transitive* if it is transitive 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 5.5. A branch evaluation $\mathcal{B}$ is called

- *splittable*[12] if every $\mathcal{JF}$-branch is totally splittable;
- *transitive* if every $\mathcal{JF}$-branch is totally transitive;
- *decided* if every $\mathcal{JF}$-branch is totally decided;
- *parametric* if $\mathcal{B}(\mathbf{b}) \in \mathcal{F}_o$ for any $\mathcal{JF}$-branch $\mathbf{b}$;

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

Lemma 5.6. *Any transitive or decided branch evaluation is splittable.*

Example 5.7. The branch evaluation $\mathcal{B}_{\mathrm{sp}}$ is decided, $\mathcal{B}_{\mathrm{KK}}$, $\mathcal{B}_{\mathrm{wf}}$ and $\mathcal{B}_{\mathrm{cwf}}$ are transitive and parametric, and $\mathcal{B}_{\mathrm{st}}$ and $\mathcal{B}_{\mathrm{cst}}$ are splittable. The branch evaluation $\mathcal{B}_{\mathrm{ex}}$ from Example 3.12 is transitive and parametric. ▲

Any branch $\mathbf{b} : x_0 \to x_1 \to \cdots$ has a least initial segment $\mathbf{b}^*$ that is decided. Indeed, if $\mathbf{b}$ is first decided at i, then $x_0 \to \cdots \to x_i$ is this segment. If $\mathbf{b}$ is nowhere decided, then $\mathbf{b}$ itself is this segment. Using this segment, we can give an equivalent characterisation of splittable.

Proposition 5.8. *A branch evaluation $\mathcal{B}$ is splittable if and only if for every branch $\mathbf{b}$, we have that the final segment of $\mathbf{b}^*$ is decided and has the same evaluation as $\mathbf{b}$.*

[12] The concept of splittable branch evaluations was first introduced in the master's thesis of the second author (Passchyn 2017).

Corollary 5.9. *Let $\mathcal{B}$ be a splittable branch evaluation. If a branch $\mathbf{b}$ is first decided at j. Then for all $0 \leq i < j$ the path $x_i \to \cdots \to x_j$ is decided and has the same evaluation as $\mathbf{b}$.*

Transitive justification systems over complementary justification frames are local in some sense.

Lemma 5.10. *Let $\mathcal{B}$ be a transitive branch evaluation such that finite branches are mapped to their last element. Let $\mathcal{JF}$ be a complementary justification frame Then for any subset I of $\mathcal{F}$ and any $x \in \mathcal{F}_d$*

- *$x \in \mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}(I)$ if and only if there is an $A \in \mathcal{JF}(x)$ such that $A \subseteq \hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I)$;*
- *$\sim x \notin \mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}(I)$ if and only if there is an $A \in \mathcal{JF}(x)$ such that $A \subseteq \mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I))$.*

The condition on finite branches in Lemma 5.10 is necessary as shown by the next example.

Example 5.11. Let $\mathcal{F}_d = \{a, \sim a\}$, $\mathcal{F}_o = \mathcal{L} \cup \{b, \sim b\}$ and $R = \{a \leftarrow b; \sim a \leftarrow \sim b\}$. Take the justification system $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ with $\mathcal{B}$ be a branch evaluation over $\mathcal{JF}$ such that $\mathcal{B} = \mathcal{B}_{\mathrm{KK}}$ on infinite branches and $\mathcal{B}(\mathbf{b}) = \sim\mathcal{B}_{\mathrm{KK}}(\mathbf{b})$ for finite branches $\mathbf{b}$. This branch evaluation is transitive and parametric. Take $I = \{\mathbf{t}, \mathbf{i}, b\}$. There exists an $A \in \mathcal{JF}(a)$ such that $A \subseteq \hat{\mathcal{S}}_{\mathcal{JS}}(I) = \{\mathbf{t}, \mathbf{i}, b, \sim a\}$, i.e. $A = \{b\}$. However, $a \notin \mathcal{S}_{\mathcal{JS}}(I) = \{\sim a\}$. ▲

6 Pasting justifications

In this section we define an operation that "glues" two justifications together. We investigate how branches are related after the gluing.

If a justification J is a subset of a justification K, then we say that K extends J or that J is a restriction of K. The union of two justifications is not necessarily a justification. However, the union of two justifications with disjoint domain is a justification.

Definition 6.1. The *extension* of J with K is defined as

$$(J \uparrow K)(x) := \begin{cases} J(x) & \text{if } x \in \mathrm{dom}(J); \\ K(x) & \text{if } x \in \mathrm{dom}(K) \setminus \mathrm{dom}(J). \end{cases}$$

It is clear that $J \uparrow K$ is an extension of J.

Branches of the extension of two locally completely justifications behave nicely.

Lemma 6.2. *If J and K are two locally complete justifications, then $J \uparrow K$ is locally complete. Moreover, every $(J \uparrow K)$-branch either is a K-branch or is a concatenation of a K-path with a J-branch. (K-path means a path in G_K)*

The next lemma illustrates the power of splittable branch evaluations.

Lemma 6.3. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a splittable justification system. Take a subset I of $\mathcal{F}$. Let J and K be two locally complete justifications that support their domain in I under $\mathcal{JS}$, then $J \uparrow K$ also supports its domain in I under $\mathcal{JS}$.*

This result also holds for non locally complete justifications.

Lemma 6.4. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a splittable justification system. Take a subset I of $\mathcal{F}$. Let J and K be arbitrary justifications that support their domain in I under $\mathcal{JS}$, then $J \uparrow K$ also supports its domain in I under $\mathcal{JS}$.*

A *root* of a graph is a node such that all other nodes can be reached with a path starting in this root. We say a justification J is *root-supported* in I if it has a root that is supported by J in I. Transitive branch evaluations act well with root-supported justifications.

Lemma 6.5. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a justification system with $\mathcal{B}$ transitive. An I-root-supported justification supports its domain in I.*

The following proposition allows us to paste together an arbitrary number of justifications that support their domain into a justification that also supports its domain.

Proposition 6.6. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a splittable justification system. Take a subset I of $\mathcal{F}$. Let X be a subset of $\mathcal{F}_d$ such that for any $x \in X$, there exists a (locally complete) justification J_x that supports its domain in I. Then there exists a (locally complete) justification J that supports its domain $\mathrm{dom}(J) = \cup_{x \in X} \mathrm{dom}(J_x)$ in I and for any $y \in \mathrm{dom}(J)$, it holds that $J(y) = J_x(y)$ for some $x \in X$.*

Proof. The proposition is trivial if $X = \emptyset$, so assume $X \neq \emptyset$. We do a proof by transfinite induction. By the well-ordering theorem[13], we can fix a well-order on $X = \{x_i \mid i \leq \beta\}$ for some ordinal β. Define

- $K_0 = J_{x_0}$;
- $K_{i+1} = K_i \uparrow J_{x_{i+1}}$ for any ordinal $i < \beta$;
- $K_\alpha = \cup_{i<\alpha} K_i$ for any limit ordinal $\alpha \leq \beta$.

The zero case trivially supports its domain and the successor case by Lemma 6.4. For a limit ordinal $\alpha \leq \beta$, take a locally complete extension L of K_α and $x \in \mathrm{dom}(K_\alpha)$. There is an $i < \alpha$ such that $x \in \mathrm{dom}(K_i)$. Since x is supported by K_i in I and L is an extension of K_α and thus also of K_i, we have that x is supported by L in I. Because L is taken arbitrary, x is supported by K_α in I; hence K_α supports its domain in I.

The condition that for $y \in \mathrm{dom}(K_i)$, it holds that $K_i(y) = J_x(y)$ for some $x \in X$ is trivial for all cases by the definition of $\uparrow$ and $\cup$.

The zero case is trivially locally complete, the successor case by Lemma 6.2. Suppose by contradiction that the limit case does not preserve locally completeness. Thus, there exist defined facts x, y such that $x \in K_\alpha(y)$, but $x \notin \mathrm{dom}(K_\alpha)$. There is an $i < \alpha$ so that $K_\alpha(y) = K_i(y)$. Then this means that x is a leaf of K_i, which is a contradiction, that is, the limit case preserves locally completeness.

Finally, set $J = K_\beta$. □

This result enables to construct a justification that support $\mathcal{S}_{\mathcal{JS}}(I)$ in I under $\mathcal{JS}$.

[13] This is equivalent to the axiom of choice.

Corollary 6.7. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a splittable justification system. For any subset I of $\mathcal{F}$, there exists a justification J that supports its domain $\mathrm{dom}(J) = \mathcal{S}_{\mathcal{JS}}(I)$ in I. Moreover, if $\mathcal{B}$ is transitive, J can be taken to be locally complete.*

If $\mathcal{B}$ is not splittable, then such a justification as in Corollary 6.7 does not necessarily exist as shown by the following example.

Example 6.8. Let $\mathcal{B}$ be the branch evaluation that maps $x_0 \to x_1 \to x_2 \to \cdots$ to x_2 if it exists and to $\sim x_1$ otherwise. Let $\mathcal{F}_d = \{a, b, \sim a, \sim b\}$ and $\mathcal{F}_o = \mathcal{L} \cup \{c, d, \sim c, \sim d\}$. Let $R = \{a \leftarrow b; b \leftarrow c; b \leftarrow d\}$. Define $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \cup R^* \rangle$ and take $I = \{\mathbf{t}, \mathbf{u}, c, \sim d\}$. Then $\mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}(I) = \{a, b\}$. However, the only connected justifications that contain a and b in its domain are

$$\begin{array}{cc} a & a \\ \downarrow & \downarrow \\ b & b \\ \downarrow & \downarrow \\ c & d \end{array}$$

These justifications do not simultaneously support a and b in I. ▲

If $\mathcal{B}$ is not transitive, then there does not necessarily exist a locally complete justification that supports its domain $\mathrm{dom}(J) = \mathcal{S}_{\mathcal{JS}}(I)$ in I. That is, the moreover part of Corollary 6.7 does not hold for non-transitive branch evaluations. This is illustrated in the following example.

Example 6.9. Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}_{\mathrm{sp}} \rangle$ be the justification system with $\mathcal{F}_d = \{a, \sim a, b, \sim b\}$, $\mathcal{F}_o = \mathcal{L} \cup \{c, \sim c\}$ $R = \{a \leftarrow b; b \leftarrow c; \sim a \leftarrow \sim b; \sim b \leftarrow \sim c\}$. Take $I = \{a, b, \sim c\}$, then $\mathcal{S}^{\mathcal{B}_{\mathrm{sp}}}_{\mathcal{JF}}(I) = \{a, \sim b\}$. Now if a locally complete justification has a in its domain, then it has b in its domain. Therefore there is no locally complete justification with domain $\mathcal{S}^{\mathcal{B}_{\mathrm{sp}}}_{\mathcal{JF}}(I)$ that supports its domain. ▲

However, we can still find a locally complete justification J that supports $\mathcal{S}_{\mathcal{JS}}(I)$ in I, i.e. any locally complete extension of a justification from Corollary 6.7.

The justification found in Corollary 6.7 is a conglomerated explanation for all supported facts and will prove to be useful later on; particularly in the proof of our main theorem.

7 Consistency of supported value

A justification that supports x in I gives an explanation why x is valid. If we have a justification for x, we expect not to find a justification for $\sim x$. Under reasonable assumptions this holds.

Proposition 7.1. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a complementary justification system with $\mathcal{B}$ consistent. Take a subset I of F. If $x \in \mathcal{F}_d$ is supported by $\mathcal{JS}$ in I, then $\sim x$ is not supported by $\mathcal{JS}$ in $\mathrm{d}(I)$.*

If I is a lower interpretable set, then Proposition 7.1 states that if the supported value of x is $\mathbf{t}$ or $\mathbf{i}$, then the supported value of $\sim x$ is not equal to $\mathbf{t}$ or $\mathbf{u}$. These two statements do not conflict each other. If I is an upper interpretable set, this is the same but $\mathbf{u}$ and $\mathbf{i}$ swapped.

Proposition 7.1 is equivalent with

$$\mathcal{S}_{\mathcal{JS}}(I) \subseteq \mathrm{d}(\mathcal{S}_{\mathcal{JS}}(\mathrm{d}(I))).$$

This in turn implies that

$$\hat{\mathcal{S}}_{\mathcal{JS}}(\mathrm{d}(I)) \subseteq \mathrm{d}(\hat{\mathcal{S}}_{\mathcal{JS}}(I)).$$

As a consequence we get a partial result concerning supported values.

Corollary 7.2. *If $\mathcal{JS}$ is a complementary justification system with a consistent branch evaluation, then*

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

for any $x \in \mathcal{F}$ and interpretation $\mathcal{I}$ of $\mathcal{F}$, and where $\leq_t$ denotes the truth order.

The converse of Proposition 7.1 does not always hold as illustrated in Example 3.12. For some branch evaluations the converse does hold.

Proposition 7.3. *Let $\mathcal{JF}$ be a complementary justification frame. Then for $\mathcal{B} \in \{\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{st}}, \mathcal{B}_{\mathrm{KK}}, \mathcal{B}_{\mathrm{wf}}\}$ it holds that if $\sim x$ is not supported in I, then x is supported in $\mathrm{d}(I)$ for all interpretable subsets I of $\mathcal{F}$.*

The proof heavily depends on Corollary 6.7 and can be found in the extended version of this text. Intuitively, if I is a lower interpretable set, Proposition 7.3 states that if the supported value of x is $\mathbf{f}$ or $\mathbf{u}$, then the supported value of $\sim x$ is $\mathbf{t}$ or $\mathbf{i}$. If I is an upper interpretable set, Proposition 7.3 states that if the supported value of x is $\mathbf{f}$ or $\mathbf{i}$, then the supported values of $\sim x$ is $\mathbf{t}$ or $\mathbf{u}$. Proposition 7.3 is equivalent with

$$\mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I)) \subseteq \hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(\mathrm{d}(I))$$

Combining Proposition 7.1 and Proposition 7.3, we get that the following theorem.

Theorem 7.4. *Let $\mathcal{JF}$ be a complementary justification frame. Let I be an interpretable subset of $\mathcal{F}$. Then $x \in \mathcal{F}_d$ is supported in I under $\mathcal{B}$ if and only if $\sim x$ is not supported in $\mathrm{d}(I)$ under $\mathcal{B}$ for $\mathcal{B} \in \{\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{KK}}, \mathcal{B}_{\mathrm{st}}, \mathcal{B}_{\mathrm{wf}}\}$. This is equivalent with $\mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I)) = \hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(\mathrm{d}(I))$.*

Corollary 7.5. *For the justification systems above, the supported value commutes with negation, i.e.*

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

for all interpretations $\mathcal{I}$ of $\mathcal{F}$.

This means that $\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \cdot)$ is an interpretation of $\mathcal{F}$. This result is what one intuitively expects. However, it is not too difficult to come up with an example in which this intuition does not hold.

Example 7.6. *Continuation of Example 3.12.* For any interpretation $\mathcal{I}$ of $\mathcal{F}$, we have that $\mathrm{SV}(\mathcal{I}, a) = \mathbf{f}$, while $\mathrm{SV}(\mathcal{I}, \sim a) = \mathbf{u}$. Therefore, $\mathrm{SV}(\mathcal{I}, \cdot)$ is not an interpretation of $\mathcal{F}$. ▲

The branch evaluation in this example induces a defective semantics since it has a counter-intuitive notion of support. Our expectation is that the supported value is always an interpretation and equal to the one corresponding to $\hat{\mathcal{S}}_{\mathcal{JS}}(I)$ as lower bound.

A similar result is already known for tree-like justifications for three-valued completion, stable and well-founded semantics, see (Denecker and De Schreye 1993).

By Corollary 4.15, we have that Theorem 7.4 also holds for $\mathcal{B}_{\text{cwf}}$ and $\mathcal{B}_{\text{cst}}$. In justification systems where the supported value commutes with negation, we have in most cases that the extended support operator preserves consistency of interpretable subsets.

Proposition 7.7. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a justification system such that $\mathrm{d}(\hat{\mathcal{S}}_{\mathcal{JS}}(I)) = \hat{\mathcal{S}}_{\mathcal{JS}}(\mathrm{d}(I))$ for all interpretable subsets I of $\mathcal{F}$. Suppose $\mathcal{B}$ does not map $\mathcal{JF}$-branches to* **i**. *Then $\hat{\mathcal{S}}_{\mathcal{JS}}$ preserves consistency of interpretable subsets.*

Remark 7.8. For a branch evaluation $\mathcal{B}$ that maps finite branches to their last element, $\mathcal{B}$ not mapping $\mathcal{JF}$-branches to **i** implies that rules cannot have **i** in their body.

Proposition 7.7 states that if an interpretation $\mathcal{I}$ is three-valued, then $\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \cdot)$ is a three-valued interpretation.

8 Conclusion

Justification theory is an abstract framework to define and study semantics of non-monotonic logics. This includes various types of logic programs, abstract argumentation, inductive and co-inductive definitions, and nested definitions. Next to being useful to define semantics of logics, the framework can also be used when solving practical problems, for instance for computing unfounded sets in modern ASP solvers (Gebser, Kaufmann, and Schaub 2012; Mariën et al. 2008), relevance checking (Jansen et al. 2016), lazy grounding algorithms (De Cat et al. 2015; Bogaerts and Weinzierl 2018), and provenance systems for databases (Damásio, Analyti, and Antoniou 2013).

Complementary justification frames arise naturally for a range of formalisms, such as, abstract argumentation and the duality between CNF and DNF formulas. In many systems, rules are only defined for atoms, such as, in logic programming, while in abstract argumentation rules are only defined for negative literals. These rules can be transformed into a complementary justification frame.

The introduced dual branch evaluations provide a systematic way to define a dual semantics in justification theory. To the extent of our knowledge, we are the first to define such duality. In particular, it shows the relation between inductive and co-inductive logic programs.

By introducing splittable branch evaluations, we have found a general class in which supporting justifications can be pasted. An interesting question is if there is a more general class of branch evaluations that satisfies Proposition 6.6 and Corollary 6.7.

This pasting is then used to prove that the supported value commutes with negation for completion, Kripke-Kleene, stable and well-founded semantics. Using our duality result, we also get it for co-stable and co-well-founded semantics. A similar result was proven for tree-like justifications in a three-valued setting in (Denecker and De Schreye 1993). An interesting question for future work is to find a complete characterisation for when the supported value commutes with negation. This would give a characterisation which branch evaluations do not induce a defective semantics.

References

Beck, H.; Eiter, T.; and Folie, C. 2017. Ticker: A system for incremental asp-based stream reasoning. *TPLP* 17(5-6):744–763.

Belnap, N. D. 1977. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*. Reidel, Dordrecht. 8–37. Invited papers from ISMVL.

Bogaerts, B., and Weinzierl, A. 2018. Exploiting justifications for lazy grounding of answer set programs. In *Proceedings of IJCAI*, 1737–1745.

Cabalar, P.; Fandinno, J.; and Fink, M. 2014. Causal graph justifications of logic programs. *TPLP* 14(4–5):603–618.

Clark, K. L. 1978. Negation as failure. In *Logic and Data Bases*, 293–322. Plenum Press.

Damásio, C. V.; Analyti, A.; and Antoniou, G. 2013. Justifications for logic programming. In Cabalar, P., and Son, T. C., eds., *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *LNCS*, 530–542. Springer.

De Cat, B.; Bogaerts, B.; Devriendt, J.; and Denecker, M. 2013. Model expansion in the presence of function symbols using constraint programming. In *Proceedings of ICTAI*, 1068–1075.

De Cat, B.; Denecker, M.; Bruynooghe, M.; and Stuckey, P. J. 2015. Lazy model expansion: Interleaving grounding with search. *J. Artif. Intell. Res. (JAIR)* 52:235–286.

Denecker, M., and De Schreye, D. 1993. Justification semantics: A unifying framework for the semantics of logic programs. In *Proceedings of LPNMR*, 365–379.

Denecker, M., and Ternovska, E. 2004. A logic of nonmonotone inductive definitions and its modularity properties. In Lifschitz, V., and Niemelä, I., eds., *LPNMR*, volume 2923 of *LNCS*, 47–60. Springer.

Denecker, M., and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Trans. Comput. Log.* 9(2):14:1–14:52.

Denecker, M., and Vennekens, J. 2007. Well-founded semantics and the algebraic theory of non-monotone inductive definitions. In *Proceedings of LPNMR*, 84–96.

Denecker, M., and Vennekens, J. 2014. The well-founded semantics is the principle of inductive definition, revisited. In *Proceedings of KR*, 22–31.

Denecker, M.; Bogaerts, B.; and Vennekens, J. 2018. Causal reasoning in a logic with possible causal process semantics. under review.

Denecker, M.; Brewka, G.; and Strass, H. 2015. A formal theory of justifications. In *Proceedings of LPNMR*, 250–264.

Denecker, M.; Marek, V.; and Truszczyński, M. 2000. Approximations, stable operators, well-founded fixpoints and

applications in nonmonotonic reasoning. In *Logic-Based Artificial Intelligence, Springer*, volume 597, 127–144.

Denecker, M.; Marek, V.; and Truszczyński, M. 2011. Reiter's default logic is a logic of autoepistemic reasoning and a good one, too. In *Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary*. College Publications. 111–144.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AIJ* 77(2):321 – 357.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *AIJ* 187:52–89.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of ICLP/SLP*, 1070–1080.

Gupta, G.; Bansal, A.; Min, R.; Simon, L.; and Mallya, A. 2007. Coinductive logic programming and its applications. In Dahl, V., and Niemelä, I., eds., *ICLP*, volume 4670 of *LNCS*, 27–44. Springer.

Harrison, A.; Lifschitz, V.; and Yang, F. 2013. On the semantics of gringo. *CoRR* abs/1312.6149.

Jansen, J.; Bogaerts, B.; Devriendt, J.; Janssens, G.; and Denecker, M. 2016. Relevance for SAT(ID). In *Proceedings of IJCAI*, 596–603.

Marek, V., and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*. Springer-Verlag. 375–398.

Mariën, M.; Wittocx, J.; Denecker, M.; and Bruynooghe, M. 2008. SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In *Proceedings of SAT*, 211–224.

Passchyn, N. 2017. Justification frames. introducing splittable branch evaluations. Master Thesis; Denecker, Marc (supervisor).

van Emden, M. H., and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *J. ACM* 23(4):733–742.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *J. ACM* 38(3):620–650.