A Formal Theory of Justifications

Marc Denecker, Gerhard Brewka, Hannes Strass
KU Leuven and U. Leipzig

September 28, 2015

Motivation and Context

Goal in KR:

- Build rich logics
- by integrating useful and expressive language constructs
- in a meaning preserving way

One motivating example

To add a nested cardinality aggregate *Card* to classical logic:

- Plug new inductive rule in definition of term:
  - $Card(\{x : \varphi\})$ is a term if φ is a formula
- Plug new inductive rule in definition of term evaluation:
  - $(Card(\{x : \varphi\})^{\mathfrak{A}} = \#(\{d \mid \mathfrak{A}[x : d] \models \varphi\})$

One motivating example

To add a nested cardinality aggregate *Card* to classical logic:

- Plug new inductive rule in definition of term:
  - $Card(\{x : \varphi\})$ is a term if φ is a formula
- Plug new inductive rule in definition of term evaluation:
  - $(Card(\{x : \varphi\}))^{\mathfrak{A}} = \#(\{d \mid \mathfrak{A}[x : d] \models \varphi\})$

We are ready.

To add aggregate expressions to logic programming and ASP:
many effort years, several PhD's and many papers.

Towards one solution

At ICLP 2015, Dasseville, Van der Hallen, Janssens, D

- Framework to add rule sets under well-founded or stable semantics to arbitrary logics with a three-valued semantics.
- Infinitely many logics can be built, including with rule sets nested in bodies.
- We used it to define a higher order logic of template definitions.

Goal of this paper

- Building generic framework for construction of logics
- Based on the basic notion of

  justification

  $\rightarrow$ a reason, a cause

Goal of this paper

- Building generic framework for construction of logics
- Based on the basic notion of

  justification

  $\rightarrow$ a reason, a cause
- An old idea: (early 90ties)
  - (Fages,91), (Pereira et al.,92), (D,Deschreye, 93)
- Recently revived:
  - (Schultz, Toni, 2013) (Cabalar, Fandinno, Fink, 2014), (Pontelli, Son, Elkhatib, 2009), (Damasio, Analyti, Antoniou, 2013)

Contributions

- A new (language independent) framework:
  - Within the same formalism: classifying different semantical principles
  - Amongst different formalisms: abstracting common semantical principles
    - e.g. logic programs vs. argumentation frameworks
  - New semantics that nobody thought of yet
  - Another form of nesting for meaning preserving integration of language constructs

Contributions

- A new (language independent) framework:
  - Within the same formalism: classifying different semantical principles
  - Amongst different formalisms: abstracting common semantical principles
    - e.g. logic programs vs. argumentation frameworks
  - New semantics that nobody thought of yet
  - Another form of nesting for meaning preserving integration of language constructs

- Applied here to argumentation theory and logic programming extensions (including coinductive logic programming (Gupta Bansal, Min, Simon, Mallya,2007))

Fact spaces

- A fact space $\mathcal{F}$
  - $\cong$ set of literals
  - positive facts $\mathcal{F}^p$, negative facts $\mathcal{F}^n$
    negation operator $\sim$: $\mathcal{F}^p \to \mathcal{F}^n$, $\mathcal{F}^n \to \mathcal{F}^p$.

Fact spaces

- A fact space $\mathcal{F}$
  - $\cong$ set of literals
  - positive facts $\mathcal{F}^p$, negative facts $\mathcal{F}^n$
    negation operator $\sim$: $\mathcal{F}^p \to \mathcal{F}^n$, $\mathcal{F}^n \to \mathcal{F}^p$.

- An interpretation $\mathfrak{A} \subseteq \mathcal{F}$
  - $\cong$ a four-valued interpretation
    - e.g., $x \in \mathfrak{A}, \sim x \notin \mathfrak{A}$: x is true
    - e.g., $x \in \mathfrak{A}, \sim x \in \mathfrak{A}$: x is inconsistent

Justification frames

A justification frame $\mathcal{JF}$:

- partitions $\mathcal{F}$ in two
  - $\mathcal{F}_d$: defined facts
  - $\mathcal{F}_o$: parameter facts
- consists of a set of rules

$$x \leftarrow S$$

  - $x \in \mathcal{F}_d, S \subseteq \mathcal{F}$, $S \neq \emptyset$
  - atomic facts are represented as $x \leftarrow \{\mathbf{true}\}$
  - negative facts in head allowed

Justifications, branches

- A justification J of justification frame $\mathcal{JF}$:
  - per defined fact $x \in \mathcal{F}_d$, choice of one rule $(x \leftarrow S) \in \mathcal{JF}$
  - defines a graph:

$$\text{arc } x \rightarrow y \quad \text{if} \quad y \in S \text{ where } (x \leftarrow S) \in J.$$

  Leafs of J are parameter facts in $\mathcal{F}_o$.

Justifications, branches

- A justification J of justification frame $\mathcal{JF}$:
  - per defined fact $x \in \mathcal{F}_d$, choice of one rule $(x \leftarrow S) \in \mathcal{JF}$
  - defines a graph:

    $$\text{arc } x \rightarrow y \quad \text{if} \quad y \in S \text{ where } (x \leftarrow S) \in J.$$

    Leafs of J are parameter facts in $\mathcal{F}_o$.

- A branch of J at x: a maximally long path $x \rightarrow x_1 \rightarrow \dots$ in J
  - either terminates in a parameter fact, or is infinite.

Branch evaluations

A branch evaluation maps branches to $\mathcal{F}$

$$\mathcal{B} : \mathit{Branches} \to \mathcal{F}$$

Branch evaluations

A branch evaluation maps branches to $\mathcal{F}$

$$\mathcal{B} : Branches \to \mathcal{F}$$

Examples:

- $\mathcal{B}_{sp}(x \to x_1 \to \dots) = x_1$ (for supported semantics)

Branch evaluations

A branch evaluation maps branches to $\mathcal{F}$

$$\mathcal{B} : Branches \to \mathcal{F}$$

Examples:

- $\mathcal{B}_{sp}(x \to x_1 \to \dots) = x_1$ (for supported semantics)
- $\mathcal{B}_{wf}(x \to x_1 \to \dots) =$
  - x_n if the branch terminates in x_n
  - **t** if the branch has a tail of negative facts
  - **f** if the branch has a tail of positive facts
  - **u** if only mixed tails and $x \in \mathcal{F}^p$
  - $\sim$**u** if only mixed tails and $x \in \mathcal{F}^n$

  (for well-founded semantics)

Branch evaluations

A branch evaluation maps branches to $\mathcal{F}$

$$\mathcal{B} : Branches \to \mathcal{F}$$

Examples:

- $\mathcal{B}_{sp}(x \to x_1 \to \dots) = x_1$ (for supported semantics)
- $\mathcal{B}_{wf}(x \to x_1 \to \dots) =$
  - x_n if the branch terminates in x_n
  - $\mathbf{t}$ if the branch has a tail of negative facts
  - $\mathbf{f}$ if the branch has a tail of positive facts
  - $\mathbf{u}$ if only mixed tails and $x \in \mathcal{F}^p$
  - $\sim\mathbf{u}$ if only mixed tails and $x \in \mathcal{F}^n$

  (for well-founded semantics)
- Others for stable semantics, Kripke-Kleene, coinductive wfs, ...

The branch evaluation is the generic parameter in the framework. By varying it, we get different semantics for the same justification frame.

Facts supported by justification in $\mathfrak{A}$

Given justification frame $\mathcal{JF}$, branch evaluation $\mathcal{B}$

- A justification J supports fact x in interpretation $\mathfrak{A}$ if:
  - $\mathcal{B}(B) \in \mathfrak{A}$, for every branch $B = (x \to \dots)$ in J

  Explanation:
  - for every branch B starting at x in J
  - apply branch evaluation $\mathcal{B}$ on B: this returns a fact
  - Check if $\mathcal{B}(B) \in \mathfrak{A}$.

Models of $\mathcal{JF}$ under $\mathcal{B}$

Given $\mathcal{JF}$, $\mathcal{B}$.

- A fact x is supported in $\mathfrak{A}$ if some justification J supports x in $\mathfrak{A}$.

Models of $\mathcal{JF}$ under $\mathcal{B}$

Given $\mathcal{JF}$, $\mathcal{B}$.

- A fact x is supported in $\mathfrak{A}$ if some justification J supports x in $\mathfrak{A}$.
- An interpretation $\mathfrak{A}$ is a model of $\mathcal{JF}$ under $\mathcal{B}$ if the set $\mathfrak{A} \cap \mathcal{F}_d$ of its defined elements is the set of supported facts in $\mathfrak{A}$.

Further more:

- Assigning an operator $T_{\mathcal{JF}}$ to justification framework (and branch evaluation $\mathcal{B}$)

Further more:

- Assigning an operator $T_{\mathcal{JF}}$ to justification framework (and branch evaluation $\mathcal{B}$)
- Easy transformations:
  - of an argumentation framework F to a justification frame $\mathcal{JF}_F$
  - of a logic program Π to a justification frame $\mathcal{JF}_\Pi$

Further more:

- Assigning an operator $T_{\mathcal{JF}}$ to justification framework (and branch evaluation $\mathcal{B}$)
- Easy transformations:
  - of an argumentation framework F to a justification frame $\mathcal{JF}_F$
  - of a logic program Π to a justification frame $\mathcal{JF}_\Pi$
- Complement Closure: deriving rules for negative facts from those of positive facts.

Theorem

Let $F = (A, X)$ be an argumentation framework and $\mathcal{JF}_F^c$ be the complement closure of its associated justification frame $\mathcal{JF}_F$. A consistent interpretation $\mathfrak{A}$

- *is stable for F iff it is an exact fixpoint of $T_{\mathcal{JF}_F^c}$*
- *is complete for F iff it is a fixpoint of $T_{\mathcal{JF}_F^c}$;*
- *is preferred for F iff it is a $\subseteq$-maximal fixpoint of $T_{\mathcal{JF}_F^c}$;*
- *is grounded for F iff it is the $\subseteq$-least fixpoint of $T_{\mathcal{JF}_F^c}$;*
- *is admissible for F iff it satisfies $\mathfrak{A} \subseteq T_{\mathcal{JF}_F^c}(\mathfrak{A})$.*

Theorem

Let Π be a logic program and $\mathcal{JF}$ be the complement closure of $\mathcal{JF}_{\Pi}$. A (4-valued) interpretation $\mathfrak{A}$

- *a supported model of Π iff $\mathfrak{A}$ is an exact model of $\mathcal{JF}$ under $\mathcal{B}_{sp}$*
- *the Kripke Kleene model of Π iff $\mathfrak{A}$ is the model of $\mathcal{JF}$ under $\mathcal{B}_{KK}$*
- *a stable model of Π iff $\mathfrak{A}$ is an exact model of $\mathcal{JF}$ under $\mathcal{B}_{st}$*
- *the well-founded model of Π iff $\mathfrak{A}$ is the model of $\mathcal{JF}$ under $\mathcal{B}_{wf}$*

Nested justification systems

- $\cong$ μ-calculus, FO(LFP)
- A nested definition $=$ a finite tree of pairs $(\mathcal{JF}, \mathcal{B})$
- Compression to reduce to unnested justification frame
- Useful to add complex language constructs to bodies

Example over finite and infinite lists

$$\{P(s) \leftarrow Q(s)\} - \mathcal{B}_{cowf}$$

$$\left\{ \begin{array}{l} Q([A|s]) \leftarrow \{P(s)\} \\ Q([B|s]) \leftarrow \{Q(s)\} \end{array} \right\} - \mathcal{B}_{wf}$$

$$\downarrow$$

compression

$$\downarrow$$

$$\{P([\underbrace{B, \ldots, B}_{\geq 0}, A|s]) \leftarrow P(s)\} - \mathcal{B}_{cowf}$$

$$\downarrow$$

$P =$ the set of $\{A, B\}$-strings with infinitely many occurrences of A

Applications

Illustrated in the paper:

- nested inductive and coinductive rule sets,
  - co-inductive logic programming + nesting
- aggregate expressions

Also done:

- complex bodies

Future work

- Application to
  - logic of causality (Bogaerts, Vennekens, D, Van den Bussche, 2014)
  - abstract dialectical frameworks (Brewka, Woltran, 2010)
  - autoepistemic and default logic
- link with Approximation Fixpoint Theory