# Negation-as-failure considered harmful

Pablo R. Fillottrani

Depto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

CACIC 2006, San Luis

1. Problem
   - LP as a knowledge representation language
   - LP with Negation-as-failure
   - NAF as a knowledge representation tool
   - Semantic Web rule interchange language
2. Proposal
   - Representing defaults without NAF
   - Syntax
   - Semantics
   - Properties
3. Conclusions and future work

# Overview



- **logic programming** (LP) is not only a programming paradigm, but also provides a good language for knowledge representation

**LP rules**

$$A_0 \leftarrow A_1, \ldots, A_n$$

- it is both simple and powerful
- simplicity: several kinds of reasoning can be formalized with LP rules
- expressive power: Turing complete if functions are allowed

- **logic programming** (LP) is not only a programming paradigm, but also provides a good language for knowledge representation

## LP rules

$$A_0 \leftarrow A_1, \ldots, A_n$$

- it is both simple and powerful
- simplicity: several kinds of reasoning can be formalized with LP rules
- expressive power: Turing complete if functions are allowed

- **logic programming** (LP) is not only a programming paradigm, but also provides a good language for knowledge representation

## LP rules

$$A_0 \leftarrow A_1, \ldots, A_n$$

- it is both simple and powerful
- simplicity: several kinds of reasoning can be formalized with LP rules
- expressive power: Turing complete if functions are allowed

- **logic programming** (LP) is not only a programming paradigm, but also provides a good language for knowledge representation

## LP rules

$$A_0 \leftarrow A_1, \ldots, A_n$$

- it is both simple and powerful
- simplicity: several kinds of reasoning can be formalized with LP rules
- expressive power: Turing complete if functions are allowed

- **logic programming** (LP) is not only a programming paradigm, but also provides a good language for knowledge representation

## LP rules

$$A_0 \leftarrow A_1, \ldots, A_n$$

- it is both simple and powerful
- simplicity: several kinds of reasoning can be formalized with LP rules
- expressive power: Turing complete if functions are allowed

# Overview



- LP is extended with **negation-as-failure** (NAF)

**general LP rules**

$$A_0 \leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n$$

- providing nonmonotonic reasoning to LP
- it is a compact way of representing defaults

- LP is extended with **negation-as-failure** (NAF)

## general LP rules

$$A_0 \leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n$$

- providing nonmonotonic reasoning to LP
- it is a compact way of representing defaults

- LP is extended with **negation-as-failure** (NAF)

**general LP rules**

$$A_0 \leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n$$

- providing nonmonotonic reasoning to LP
- it is a compact way of representing defaults

- LP is extended with **negation-as-failure** (NAF)

**general LP rules**

$$A_0 \leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n$$

- providing nonmonotonic reasoning to LP
- it is a compact way of representing defaults

# Overview



- NAF is considered sometimes like negation, sometimes like epistemic operator
- it has context-sensitive semantics, so adding a new rule about apparently unrelated predicates may affect the meaning of NAF literals
- there is no symmetry between positive and negative information

- NAF is considered sometimes like negation, sometimes like epistemic operator
- it has context-sensitive semantics, so adding a new rule about apparently unrelated predicates may affect the meaning of NAF literals
- there is no symmetry between positive and negative information

- NAF is considered sometimes like negation, sometimes like epistemic operator
- it has context-sensitive semantics, so adding a new rule about apparently unrelated predicates may affect the meaning of NAF literals
- there is no symmetry between positive and negative information

- since NAF is not really negation, strong negation was also necessary to add to LP rules

**extended LP rules**

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_m, not \pm A_{m+1}, \ldots, not \pm A_n$$

- there is no relation between NAF and strong negation

- since NAF is not really negation, strong negation was also necessary to add to LP rules

**extended LP rules**

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_m, not \pm A_{m+1}, \ldots, not \pm A_n$$

- there is no relation between NAF and strong negation

- since NAF is not really negation, strong negation was also necessary to add to LP rules

**extended LP rules**

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_m, not \pm A_{m+1}, \ldots, not \pm A_n$$

- there is no relation between NAF and strong negation

## Summary

extended LP language becomes too complex

- NAF have several formal semantics
- NAF have several intended meanings
- it is represented as negation, but has no relation with negative connectives

## Summary

extended LP language becomes too complex

- NAF have several formal semantics
- NAF have several intended meanings
- it is represented as negation, but has no relation with negative connectives

## Summary

extended LP language becomes too complex

- NAF have several formal semantics
- NAF have several intended meanings
- it is represented as negation, but has no relation with negative connectives

## Summary

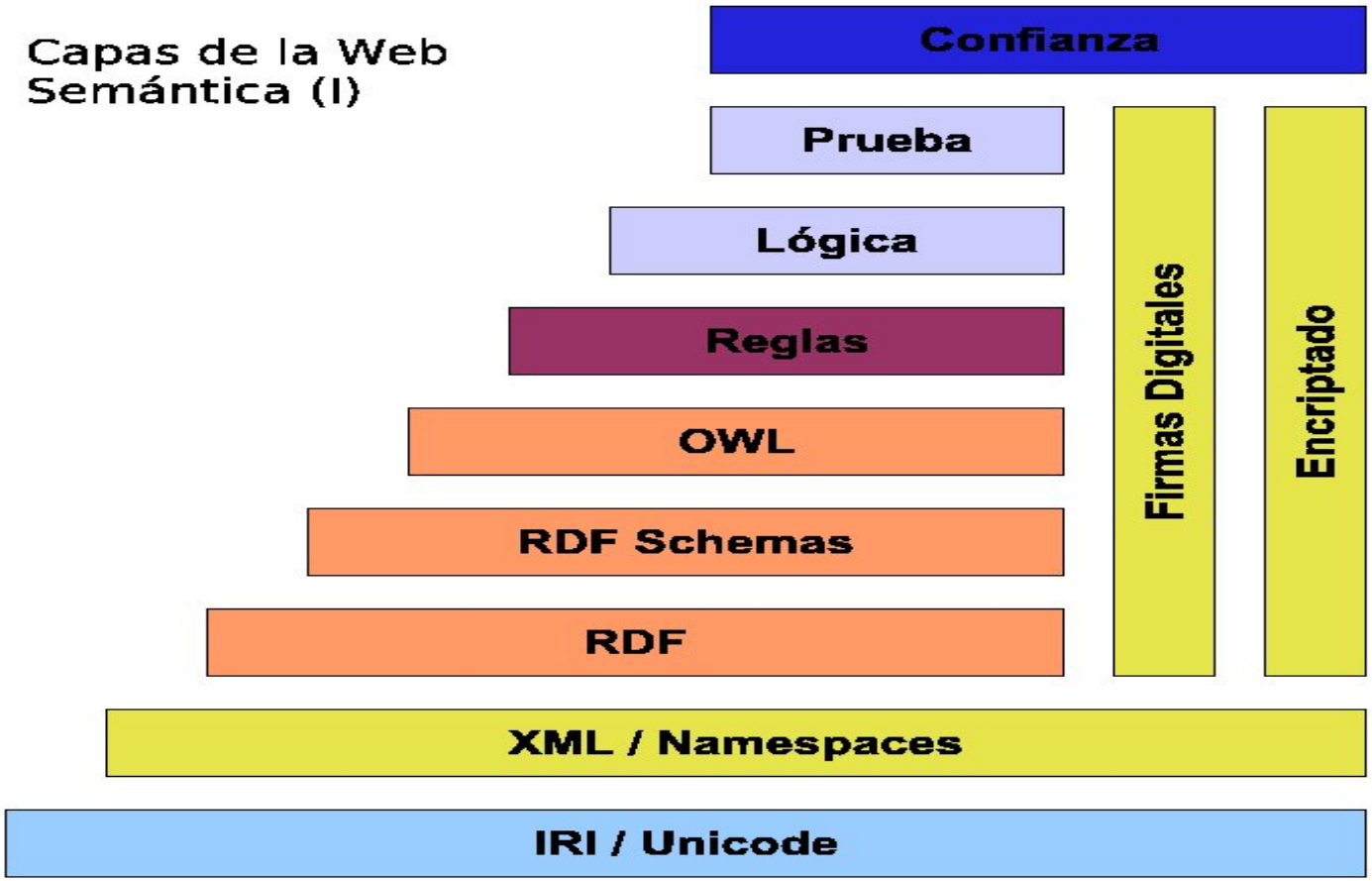
extended LP language becomes too complex

- NAF have several formal semantics
- NAF have several intended meanings
- it is represented as negation, but has no relation with negative connectives

# Overview



- rules with defaults are being integrated into the Semantic Web framework



# Overview



- other nonmonotonic formalisms (default logic, circumscription, autoepistemic logics) do not make this strong association between default inference rules and negation
- so we propose a LP language without NAF but including nonmonotonic reasoning
- strong negation is the only negative connective in the language
- integration is done by mixing the styles in circumscriptive theories and default logic

- other nonmonotonic formalisms (default logic, circumscription, autoepistemic logics) do not make this strong association between default inference rules and negation
- so we propose a LP language without NAF but including nonmonotonic reasoning
- strong negation is the only negative connective in the language
- integration is done by mixing the styles in circumscriptive theories and default logic

- other nonmonotonic formalisms (default logic, circumscription, autoepistemic logics) do not make this strong association between default inference rules and negation
- so we propose a LP language without NAF but including nonmonotonic reasoning
- strong negation is the only negative connective in the language
- integration is done by mixing the styles in circumscriptive theories and default logic

- other nonmonotonic formalisms (default logic, circumscription, autoepistemic logics) do not make this strong association between default inference rules and negation
- so we propose a LP language without NAF but including nonmonotonic reasoning
- strong negation is the only negative connective in the language
- integration is done by mixing the styles in circumscriptive theories and default logic

# Overview



- a second order predicate $\texttt{def}(\cdot)$ is added to the language
- for example,

$$\texttt{def}(\texttt{delayed}(\texttt{flight123})) \leftarrow$$

represents that $\texttt{delayed}(\texttt{flight123})$ is being considered a default fact

- these new atoms can be used anywhere in rules

LP rules with default policies

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_n$$

- any atom $A_i$ may contain the $\texttt{def}(\cdot)$ predicate

$$\texttt{def}(\texttt{onTime}(\texttt{X})) \leftarrow -\texttt{flight}(\texttt{X}, \textit{aerolineas})$$

- a second order predicate $\texttt{def}(\cdot)$ is added to the language
- for example,

$$\texttt{def}(\texttt{delayed}(\texttt{flight123})) \leftarrow$$

represents that $\texttt{delayed}(\texttt{flight123})$ is being considered a default fact

- these new atoms can be used anywhere in rules

LP rules with default policies

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_n$$

- any atom $A_i$ may contain the $\texttt{def}(\cdot)$ predicate

$$\texttt{def}(\texttt{onTime}(\texttt{X})) \leftarrow -\texttt{flight}(\texttt{X}, \textit{aerolineas})$$

- a second order predicate $\texttt{def}(\cdot)$ is added to the language
- for example,

$$\texttt{def}(\texttt{delayed}(\texttt{flight123})) \leftarrow$$

represents that $\texttt{delayed}(\texttt{flight123})$ is being considered a default fact

- these new atoms can be used anywhere in rules

**LP rules with default policies**

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_n$$

- any atom $A_i$ may contain the $\texttt{def}(\cdot)$ predicate

$$\texttt{def}(\texttt{onTime}(\texttt{X})) \leftarrow -\texttt{flight}(\texttt{X}, \textit{aerolineas})$$

- a second order predicate $\texttt{def}(\cdot)$ is added to the language
- for example,

$$\texttt{def}(\texttt{delayed}(\texttt{flight123})) \leftarrow$$

represents that $\texttt{delayed}(\texttt{flight123})$ is being considered a default fact

- these new atoms can be used anywhere in rules

**LP rules with default policies**

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_n$$

- any atom $A_i$ may contain the $\texttt{def}(\cdot)$ predicate

$$\texttt{def}(\texttt{onTime}(\texttt{X})) \leftarrow -\texttt{flight}(\texttt{X}, \textit{aerolineas})$$

- a second order predicate $\texttt{def}(\cdot)$ is added to the language
- for example,

$$\texttt{def}(\texttt{delayed}(\texttt{flight123})) \leftarrow$$

represents that $\texttt{delayed}(\texttt{flight123})$ is being considered a default fact

- these new atoms can be used anywhere in rules

## LP rules with default policies

$$\pm A_0 \leftarrow \pm A_1, \ldots, \pm A_n$$

- any atom $A_i$ may contain the $\texttt{def}(\cdot)$ predicate

$$\texttt{def}(\texttt{onTime}(\texttt{X})) \leftarrow -\texttt{flight}(\texttt{X}, \textit{aerolineas})$$

# Overview



- both answer sets and well founded model semantics can be stated in this framework
- The set of default literals of $\Pi$ with respect to $C$ is the set

$$\mathrm{Def}_{\Pi}(C) := \{L : \mathtt{def}(L) \in \mathbf{Cn}(\Pi) \wedge \bar{L} \notin C\}$$

- $S$, a set of literals, is called an answer set of $\Pi$ if it satisfies

$$S = \mathbf{Cn}(\Pi \cup \mathrm{Def}_{\Pi \cup S}(S))$$

- both answer sets and well founded model semantics can be stated in this framework
- The set of default literals of $\Pi$ with respect to $C$ is the set

$$\mathrm{Def}_{\Pi}(C) := \{L : \mathtt{def}(L) \in \mathbf{Cn}(\Pi) \wedge \bar{L} \notin C\}$$

- $S$, a set of literals, is called an answer set of $\Pi$ if it satisfies

$$S = \mathbf{Cn}(\Pi \cup \mathrm{Def}_{\Pi \cup S}(S))$$

- both answer sets and well founded model semantics can be stated in this framework
- The set of default literals of $\Pi$ with respect to $C$ is the set

$$\mathrm{Def}_{\Pi}(C) := \{L : \mathtt{def}(L) \in \mathbf{Cn}(\Pi) \wedge \bar{L} \notin C\}$$

- $S$, a set of literals, is called an answer set of $\Pi$ if it satisfies

$$S = \mathbf{Cn}(\Pi \cup \mathrm{Def}_{\Pi \cup S}(S))$$

# Overview



- we get the same behaviour as in extended logic programs, for answer sets semantics (except for reduction)
- our approach has the same expressive power as extended logic programs
- but we can observe a more disciplined use of NAF that in the translated extended logic program
- there is a clear separation between the rules that determine an atom truth value (producers), and those that use them (consumers)

- we get the same behaviour as in extended logic programs, for answer sets semantics (except for reduction)
- our approach has the same expressive power as extended logic programs
- but we can observe a more disciplined use of NAF that in the translated extended logic program
- there is a clear separation between the rules that determine an atom truth value (producers), and those that use them (consumers)

- we get the same behaviour as in extended logic programs, for answer sets semantics (except for reduction)
- our approach has the same expressive power as extended logic programs
- but we can observe a more disciplined use of NAF that in the translated extended logic program
- there is a clear separation between the rules that determine an atom truth value (producers), and those that use them (consumers)

- we get the same behaviour as in extended logic programs, for answer sets semantics (except for reduction)
- our approach has the same expressive power as extended logic programs
- but we can observe a more disciplined use of NAF that in the translated extended logic program
- there is a clear separation between the rules that determine an atom truth value (producers), and those that use them (consumers)

- we presented a framework for replacing NAF in logic programs with the introduction of default policies
- clear separation of the roles of negation and nonmonotonic inferences
- this is desirable for the future rule interchange language standard of W3C
- we are developing a DLV front-end processor to take advantage of existing provers

- we presented a framework for replacing NAF in logic programs with the introduction of default policies
- clear separation of the roles of negation and nonmonotonic inferences
- this is desirable for the future rule interchange language standard of W3C
- we are developing a DLV front-end processor to take advantage of existing provers

- we presented a framework for replacing NAF in logic programs with the introduction of default policies
- clear separation of the roles of negation and nonmonotonic inferences
- this is desirable for the future rule interchange language standard of W3C
- we are developing a DLV front-end processor to take advantage of existing provers

- we presented a framework for replacing NAF in logic programs with the introduction of default policies
- clear separation of the roles of negation and nonmonotonic inferences
- this is desirable for the future rule interchange language standard of W3C
- we are developing a DLV front-end processor to take advantage of existing provers