Argumentation in Intelligent Agents: Theory and Applications

Carlos Iván Chesñevar

cic@cs.uns.edu.ar

CONICET and Laboratory of R&D in A.I.
Department of Computer Science
Universidad Nacional del Sur
Bahía Blanca, Argentina

DeLP Web

DeLP

DeLP

Part 2 - Outline

- Introduction to Defeasible Logic Programming (DeLP).
- Application of DeLP: User Support Systems
- Applications of DeLP: Web-Based Forms.
- Extensions of DeLP: P-DeLP, O-DeLP

DeLP

DeLP

Defeasible Logic Programming: DeLP

A Defeasible Logic Program (dlp) is a set of facts, strict and defeasible rules denoted \( P = (\Pi, \Delta) \)

\[
\Pi = \begin{cases} 
\text{bird}(X) \leftarrow \text{chicken}(X) & \text{chicken(tina)} \\
\text{bird}(X) \leftarrow \text{penguin}(X) & \text{penguin(opus)} \\
\neg \text{flies}(X) \leftarrow \text{penguin}(X) & \text{scared(tina)} 
\end{cases} \quad \text{Facts}
\]

\[
\Delta = \begin{cases} 
\neg \text{flies}(X) \leftarrow \text{bird}(X) \\
\neg \text{flies}(X) \leftarrow \text{chicken}(X) \\
\neg \text{flies}(X) \leftarrow \text{chicken}(X), \text{scared}(X)
\end{cases} \quad \text{Defeasible Rules}
\]

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP

DeLP
Counter-argument

Π∪{risk(acme),¬risk(acme)} is a contradictory set

Argument Comparison: Generalized Specificity

Def: Let \( P = (\Pi, \Delta) \) be a program, let \( \Pi_0 \) be the set of strict rules in \( \Pi \) and let \( \mathcal{F} \) be the set of all literals that can be defeasibly derived from \( P \). Let \( \langle A_1, L_1 \rangle \) and \( \langle A_2, L_2 \rangle \) be two arguments built from \( P \), where \( L_1, L_2 \in \mathcal{F} \). Then \( \langle A_1, L_1 \rangle \) is strictly more specific than \( \langle A_2, L_2 \rangle \) if:

1. For all \( \mathcal{H} \subseteq \mathcal{F} \), if there exists a defeasible derivation \( \Pi_0 \cup \mathcal{H} \cup A_1 \vdash L_1 \) while \( \Pi_0 \cup \mathcal{H} \not\vdash L_1 \) then \( \Pi_0 \cup \mathcal{H} \cup A_2 \not\vdash L_2 \) and

2. There exists \( \mathcal{H} \subseteq \mathcal{F} \) such that there exists a defeasible derivation \( \Pi_0 \cup \mathcal{H} \cup A_1 \vdash L_2 \) and \( \Pi_0 \cup \mathcal{H} \not\vdash L_2 \) but \( \Pi_0 \cup \mathcal{H} \cup A_1 \not\vdash L_1 \)

(Adapted from David L. Poole’s (1985) “On the Comparison of Theories: Preferring the Most Specific Explanation,” pages 144–147 of Proceedings of 9th IJCAI.)

Deformers

An argument \( (B, P) \) is a defeater for \( (A, L) \) if \( (B, P) \) is a counter-argument \( (A, L) \) that attacks a subargument \( (S, Q) \) of \( (A, L) \) and one of the following conditions holds:

(a) \( (B, P) \) is better than \( (S, Q) \) (proper defeater), or

(b) \( (B, P) \) is not comparable to \( (S, Q) \) (blocking defeater).

Argumentation Line

Given \( P = (\Pi, \Delta) \), and \( \langle A_0, L_0 \rangle \) an argument obtained from \( P \). An argumentation line for \( \langle A_0, L_0 \rangle \) is a sequence of arguments obtained from \( P \), denoted \( \Lambda = \langle \langle A_0, L_0 \rangle, \langle A_1, L_1 \rangle, \ldots \rangle \) where each element in the sequence \( \langle A_i, L_i \rangle \), \( i \geq 0 \) is a defeater for \( \langle A_{i+1}, L_{i+1} \rangle \).

Argumentation Line

Given an argumentation line \( \Lambda = \langle \langle A_0, L_0 \rangle, \langle A_1, L_1 \rangle, \ldots \rangle \), the subsequence \( \Lambda_S = \langle \langle A_0, L_0 \rangle, \langle A_2, L_2 \rangle, \ldots \rangle \) contains supporting arguments and \( \Lambda_I = \langle \langle A_1, L_1 \rangle, \langle A_3, L_3 \rangle, \ldots \rangle \) are interfering arguments.

Argumentation Line

Given an argumentation line \( \Lambda = \langle \langle A_0, L_0 \rangle, \langle A_1, L_1 \rangle, \ldots \rangle \), the subsequence \( \Lambda_S = \langle \langle A_0, L_0 \rangle, \langle A_2, L_2 \rangle, \ldots \rangle \) contains supporting arguments and \( \Lambda_I = \langle \langle A_1, L_1 \rangle, \langle A_3, L_3 \rangle, \ldots \rangle \) are interfering arguments.
Acceptable Argumentation Line

Given a program \( P = (\Pi, \Delta) \), an argumentation line \( \Lambda = ([A_0, L_0], [A_1, L_1], \ldots) \) will be acceptable if:

1. \( \Lambda \) is a finite sequence.
2. The sets \( \Lambda_S \) of supporting arguments is concordant, and the set \( \Lambda_I \) of interfering arguments is concordant.
3. There is no argument \( [A_i, L_i] \) in \( \Lambda \) that is a subargument of a preceding argument \( [A_j, L_j] \), \( i < j \).
4. For all \( i \), such that \( [A_i, L_i] \) is a blocking defeater for \( [A_i-1, L_{i-1}] \), if there exists \( [A_i+1, L_{i+1}] \) then \( [A_i+1, L_{i+1}] \) is a proper defeater for \( [A_i, L_i] \) (i.e., \( [A_i, L_i] \) could not be blocked).

Dialectical Tree

Given a program \( P = (\Pi, \Delta) \), a literal \( L \) will be warranted if there is an argument \( [A, L] \) built from \( P \), and that argument has a dialectical tree whose root node is marked \( U \).

That is, argument \( [A, L] \) is an argument for which all the possible defeaters have been defeated.

We will say that \( A \) is a warrant for \( L \).

Answers in DeLP

Given a program \( P = (\Pi, \Delta) \), and a query for \( L \) the possible answers are:

- **YES**, if \( L \) is warranted.
- **NO**, if \( \neg L \) is warranted.
- **UNDECIDED**, if neither \( L \) nor \( \neg L \) are warranted.
- **UNKNOWN**, if \( L \) is not in the language of the program.

Strict knowledge

Defeasible knowledge

Sample rules:

- When there is pump clog, fuel is not ok:
  \( \neg fuel_{ok} \leftarrow pump_{clog} \)

- When there is heat, usually engine is not ok.
  \( \neg engine_{ok} \leftarrow heat \)

Engine has 3 switches on
There is heat...
Is the engine ok?
Is there an argument for engine_ok?

Can <A₁, engine_ok> be attacked?

Can <A₂, ~fuel_ok> be attacked?

Can <A₃, ~low_speed> be attacked?

Can <A₁, engine_ok> be attacked?

Dialectical Tree: labelling
How DeLP works

DeLP Program P
DeLe Interpreter
Abstract Machine

DeLP Interpreter

DeLP Output

Computes dialectical tree by backward chaining. Primitives are based on an extension of Warren’s abstract machine for Prolog

DeLP Development Environment

Possible Answers

- YES (there exists a warranted argument <A,q>)
- NO (there exists a warranted argument for <A,~q>)
- UNDEIDED (none of the above cases hold)
- UNKNOWN (q is not in the program signature).

Possible Answers

- YES (there exists a warranted argument <A,h>)
- NO (there exists a warranted argument for <A,~h>)
- UNDEIDED (none of the above cases hold).

DeLP: applications

Application of DeLP: User Support Systems

The slides that follow are based on the articles:


User Support Systems (USS)

- USS address the problem of Information overload by providing guidelines or hints (e.g. for buying books, plane tickets, etc.)
- USS create a model of the user’s preferences or the user’s tasks to help identify worthwhile stuff (news, web pages, books, etc.)

Goals

- Find what users want.
- Know what users like.
- Gain trustworthiness from users!

Two major kinds of USS

- Critics (or critiquing systems) are cooperative tools that observe the user interacting with a computer system, and present reasoned opinions about a product under development.
- Recommenders (or recommender systems) are programs that create a model of the user’s preferences or the user’s task with the purpose of facilitating access to items (e.g. news, Web pages, etc.) that the user may find useful.

Part 2 - Outline

- Introduction to Defeasible Logic Programming (DeLP).
- Application of DeLP: User Support Systems
- Applications of DeLP: Web-Based Forms.
- Extensions of DeLP: P-DeLP, O-DeLP
Critiquing systems in word processing

- Critics for word processing have helped to reduce the overall amount of burden for checking documents.
- Hard problem: appropriateness of language usage (e.g. some colloquial words in Spain are not allowed as such in Argentina).
- One possible solution: concordance programs, which use the Web as a repository (e.g. http://www.concordancesoftware.co.uk/)
- Problem: they require to provide search strings, having limited functionality. It would be interesting to have inference abilities...

- Argumentation + critiquing systems!

Critiquing systems in word processing

- Critics for word processing have helped to reduce the overall amount of burden for checking documents.
- Hard problem: appropriateness of language usage (e.g. some colloquial words in Spain are not allowed as such in Argentina).
- One possible solution: concordance programs, which use the Web as a repository (e.g. http://www.concordancesoftware.co.uk/)
- Problem: they require to provide search strings, having limited functionality. It would be interesting to have inference abilities...

- Our proposal: Argumentation + critiquing systems

Recommender systems

They rely on similarity measures between users or contents, computed on the basis of methods coming from IR or ML communities.

Two main techniques to compute recommendations:

- Content-Based Recommenders: infer preferences of individual users based on what the user liked in the past. They are based on the premise that user’s preferences tend to persist through time.

- Collaborative Filtering Recommenders: infer preferences of individual users based on behavior of multiple users. They are based on the assumption that users’ preferences are correlated.

- Hybrid Recommenders: combine both.

Limitations of Traditional Views for RS

- Mostly unable to perform qualitative inference on the recommendations.
- Mostly unable to deal with the defeasible nature of user’s preferences.
- Unable to provide explanations: trustworthiness issues!

Our proposal: Argumentation + recommender systems

Hybrid RS: outline

- Content-based search results
- Active user’s profile as list of rated items
- Pool of users’ profiles as list of rated items
- Collaborative filtering

- Active user information needs
- Declared interests
- Feedback
- Monitoring

- Pool of users
- Declared interests
- Feedback
- Monitoring

- Recommendations

Proposal: Model the users’ preference criteria in terms of a DeLP program built on top of a content-based search engine.

DeLP Program \( P \) involving:

- Users’ preference criteria
- P-user preferences and behavior of active user
- P-pool preferences and behavior of pool of users
- P-domain: domain background knowledge
Prioritizing Recommendations

- Recommendations can be abstracted as a list \( L = \{s_1, s_2, \ldots, s_n\} \) prioritizing its elements according to their epistemic status:
  - \( S^w \): warranted results: those results \( s_i \) for which there exists at least one warranted argument supporting the relevance of \( s_i \) wrt \( P \)
  - \( S^d \): defeated results: those results \( s_i \) for which there exists at least one warranted argument supporting the non-relevance of \( s_i \) wrt \( P \)
  - \( S^u \): undecided results: results which are neither warranted nor defeated.

High-Level Algorithm RecommendOnQuery

Input: Query \( Q \), DeLP program \( P = P_{user} \cup P_{pool} \cup P_{domain} \)

Output: List \( L_{new} \)

Begin

\[ L = \{s_1, s_2, \ldots, s_k\} \quad \text{(output of solving \( Q \) wrt content-based search engine)} \]

\[ P_{search} = \{\text{facts encoding info}(s_1), \text{info}(s_2), \ldots, \text{info}(s_k)\} \]

\[ P' = \text{Revise} \left( P \cup P_{search} \right) \]

Initialize \( S^w, S^d, S^u \) as empty sets

FOR EVERY \( s_i \) in \( L \)

DO

Solve query “rel(\( s_i \))” using DeLP program \( P' \)

IF \( \text{rel}(s_i) \) is warranted wrt \( P' \) THEN add \( s_i \) to \( S^w \)

ELSE

IF \( \neg\text{rel}(s_i) \) is warranted wrt \( P' \) THEN add \( s_i \) to \( S^d \)

ELSE add \( s_i \) to \( S^u \)

Return \( L_{new} = \{\text{warranted } s_i's, \text{ undecided } s_i's, \text{ defeated } s_i's\} \)

Case-Study: Solving Web Search Queries

- Consider a journalist who wants to search for news articles about global warming.

Google

News about global warming

Too many results!

Proposal: rank search results according to their epistemic status wrt a DeLP program representing user’s knowledge

From search results into DeLP facts

- Facts are extracted from HTML or XML tags in Google search results.
Applying Implicit Knowledge

- Articles written by Bob Clean are reliable.
- Usually, if the journalist is trustworthy then the article is reliable.
- Old articles from trustworthy journalists are not reliable (as they are out of date).
- If a journalist never faked a report then she is reliable.
- Newspapers from industrialized countries usually offer a biased viewpoint on global warming
  • "The Environmentalist" is a newspaper from an industrialized country which is non-biased.
- Tim Greenhouse faked a report.

Warming: Worked example

**Defeasible Rules**

- `rel(X) ← author(X,A),trust(A).`
- `¬rel(X) ← author(X,A),trust(A), outdated(X).`
- `trust(A) ← not faked_news(A).`
- `¬rel(X) ← address(X,Url),biased(Url).`
- `biased(Url) ← industrialized_country(Url).`
- `¬biased(Url) ← industrialized_country(Url), domain(Url,D), D="env.org/...".`

**Strict Rules**

- `Industrialized_country(X) ← [Computed elsewhere]`
- `Domain(Url,D), D="env.org/...".`

Warming: worked example

**DeLP rules**

Distinguished predicate "rel(s_i)"

- List first all results warranted to be relevant;
- Then all those whose relevance is undecided;
- Finally, all those which are warranted to be non-relevant

**Sorting search results**

For every search result `s_i` analyze relevance of `s_i` wrt implicit knowledge

**DeLP program**

S_1: Is this Article Relevant?

```
author(s1, tim_gr) address(s1, "env.org/...") date(s1,20070203) S_1
```

Status of `rel(s1)`: not warranted
Status of `¬rel(s1)`: not warranted
Status of `s1`: Undecided

S_1: Is this Article Relevant? (cntd)

```
author(s1, tim_gr) address(s1, "env.org/...") date(s1,20070203)
```

```
indus("env.org/...") domain("env.org/...", "env.org/...") ("env.org/..." = "env.org")
```

```
Status of rel(s1): not warranted
Status of ¬rel(s1): not warranted
Status of s1: Undecided
```
Using The Web as a Corpus

- A huge amount of sentences in natural languages are available as Web documents.
- Pattern matching capabilities of current search engines allow to use the Web as a linguistic corpus, reflecting the current status of a living language (e.g., English, Spanish, etc.).
- We developed an argumentative approach to provide proactive assistance for language usage assessment on the basis of the Web Corpus
- We refer to such systems as critiquing systems

The Web as a Corpus: Advantages

- Updated and free information: building up large linguistic corpora requires considerable effort; the Web corpus exists and it is free.
- State-of-the-art linguistic database: The Web corpus reflects the current status of the language, as Web documents are created, updated, and eventually deleted.
- User-friendly handling of documents: several web-based applications have been developed for effective pattern matching, clustering, text classification, etc.

Underlying concept: usage indices

- We defined several “usage indices”, which establish statistical measures associated with natural language terms. E.g:
  - Given a domain \(d\), we write \(|d|\) to denote the number of Web pages found in domain \(d\).
  - Constrained usage \(U_d(s,D)\) = how many times a given string \(s\) appears in set \(D\) of web domains.

\[
U_d(s) = |\text{Web}|
\]
\[
U_c(s,D) = |D|_s = \sum_{i=1}^{|D|} d_i
\]

Note: all these indices can be easily computed from hit counts associated with different Google queries.
Underlying concept: usage indices

- We defined several "usage indices", which establish statistical measures associated with natural language terms. E.g:
  - Given a domain d, we write ||d|| to denote the number of Web pages found in domain d.
  - Constrained usage $U_c(s,D)$ is how many times a given string $s$ appears in set $D$ of web domains.

\[
U_c(s,D) = \frac{\sum_{i=1}^{k} ||d_i||}{||D||}
\]

Note: all these indices can be easily computed from hit counts associated with different Google queries.

Usage indices: examples

Consider the strings "rearing children", "parents", and "of twins", and domains ".uk" and ".babycentre.co.uk". Then it holds that:

\[
U_c("rearing children",\{\text{.uk}\}) = 435
\]
\[
U_c("rearing children",\text{Web}) = 13700
\]
\[
U_c("rearing children",\{\text{.uk}\},\text{Web}) = 3.76
\]
\[
U_p("parents", "of twins",\{\text{.babycentre.co.uk}\}) = \frac{677}{747} = 0.906
\]

Note: the above computations are time-dependent (i.e. depend on the current Web corpus)

Assessing Natural Language Usage

- Usage indices provide statistical measures which are relevant to assess natural language usage.
  E.g: in English, "warm" and "hot" have similar meanings…
  However, "warm regards" is pragmatically valid in English… whereas "hot regards" is not !!!

Studying language usage phenomena

Several language usage phenomena can be analyzed with usage indices, e.g.

1. Analysis of calque or mimetism in non-native English speakers.
2. Dialectical usage of language
3. The scope of common usage-related phenomena.
Calque d’expression or mimetism

Consider the English verb to associate which requires the preposition with.
Consider the equivalent Spanish verb asociar, which can use prepositions a or con
Common error in English for Spanish speakers: ‘associate to’

\[ \begin{align*}
U_{rel}(s_1, s_2, \text{Web}) &= \left( \frac{U(s_1, D) + 1}{U(s_2, D) + 1} \right) \\
U_{rel}(s_1, s_2, \text{.es}) &= 0.18 \\
U_{rel}(s_1, s_2, \text{.au}) &= 0.006
\end{align*} \]

Assessing Dialectical Usage

In Spain “A qué esperas...?” is commonly used in everyday language, whereas in Argentina that sentence is uncommon. Such situations occur also in other languages...

How to assess this situation? → Usage indices

\[ \begin{align*}
U_i(a, \text{que esperas, ',ar'}) &= 632/22000 = 0.03 \\
U_i(a, \text{que esperas, ',es'}) &= 15400 / 17400 = 0.89 \\
U_i(\text{merece, la pena, ',ar'}) &= 11/399 = 0.03 \\
U_i(\text{vale, la pena, ',es'}) &= 176 / 402 = 0.44
\end{align*} \]

Application Area

Consider an American journalist who writes articles in Spanish about Latin American issues intended for audiences both in Spain and Argentina...
Spanish is not her mother tongue... so she is in doubt about the usage of certain idiomatic expressions which could lead to misunderstandings, since Spanish usage in Spain and Argentina differ (as English usage in UK and USA do)

Corralito = little baby crib (only in Argentina)

A Sample Problematic Paragraph

“El corralito fue un fenómeno muy complejo [...]
Para el colectivo de los trabajadores autónomos fue particularmente [...]

“El “corralito” was a very complex phenomenon [...] For the union of autonomous workers it was particularly [...].”

Corralito = no meaning for Spanish audience. But the term became of common usage through Spanish newspapers.
Colectivo + Prep. = “union” (but only used in Spain)

Idea: using DeLP to encode knowledge and reason about language usage, using the Web as linguistic corpus.

Control rules for language usage assessment

\[ \begin{align*}
\text{Solve}(S) &\leftarrow \text{acc}(S), \text{write}(\text{Acceptable}) \\
\text{Solve}(S) &\leftarrow \neg \text{acc}(S), \text{write}(\text{Not acceptable}). \\
\text{Solve}(S) &\leftarrow \neg \text{acc}(S), \text{repair}(S,R), \text{acc}(R), \text{write}(\text{Accept. if rephrased as }, R). \\
\text{Repair}(S,R) &\leftarrow <\text{replaces parts of } S \text{ by synonyms or related expressions using e.g. WordNet, given } R \text{ as an output} > \\
\text{Acc}(S) &\text{ holds iff } S \text{ is a string that can be warranted to be accepted according to a DeLP program containing user-specified criteria}
\end{align*} \]
De defeasible rules for language usage

1) An expression S in Spanish is usually acceptable when it is commonly used.
2) An expression S is usually not acceptable if it is not commonly used.
3) Regionalisms for Argentina are usually not acceptable.
4) Regionalism for Spain are usually not acceptable.
5) An expression S in Spanish is usually a regionalism if it is frequently used in a particular country, but not in others.
6) Expressions that are frequently used in Argentina but have gained widespread use in Spanish media are usually not regionalisms.

Predicates based on Usage Indices

rare_in_spanish(S) ← not common_in_spanish(S).
common_in_spanish(S) ← spanish_speaking(Cs),
V is Uc(S,Cs), V>200.
appears_in_news(S,C) ← news_domains(Ds,C),
V is Uc(S,Ds), V>100.
loocly_freq(S,[‘.ar’]) ← V is Ua(S,[‘.ar’],[‘.es’]), V > 10.
loocly_freq(S,[‘.es’]) ← V is Ua(S,[‘.es’],[‘.ar’]), V > 10.
spanish_speaking([‘.es’],[‘.ar’]).
news_domains([‘elmundo.es’],‘elpais.es’),[‘.es’].

Is Expression s1 Acceptable?

acc(s1) ← common_in_spanish(s1), regionalism(s1,’.ar’).
~acc(s1) ← common_in_spanish(s1), regionalism(s1,’.es’).
regionalism(S,Ctry) ← locally_freq(S,Ctry).
~regionalism(S,[‘.ar’]) ← locally_freq(S,[‘.ar’]),
appears_in_news(S,’.es’).

Is Expression s2 Acceptable?

collectivo de los trabajadores autónomos

acc(s2) ← common_in_spanish(s2),
localy_freq(s2,[‘.es’],[‘.ar’]),
~acc(s2) ← common_in_spanish(s2),
localy_freq(s2,[‘.es’],[‘.ar’]),
spanish_speaking([‘.es’],[‘.ar’]).
news_domains([‘elmundo.es’],‘elpais.es’),[‘.es’].

Critiquing System: outline
Introduction and motivations

- Forms have long been used since the beginning of the World Wide Web, and have evolved along with new markup languages (e.g. XML).
- Performing validations on form field values usually consist of hard-coded decision criteria, presented as scripts in some imperative language.
- However, there are features or attributes which may be seen as part of the “intended meaning” of a form, without being values by themselves (e.g. “reliable client”).
- The value of such attributes may change in presence of new information; we call them defeasible attributes.
- Proposal: an extension to web-based forms which allows the specification of such defeasible attributes using a defeasible knowledge base expressed in DeLP.

A Worked Example

- An international bank keeps track of its clients’ data in order to concede loans. The account manager has a number of criteria for conceding loans to clients, which correspond to a “reasonable” profile according to the clients’ personal records.

Is John a good candidate?

There is an argument \( A_1 \) for candidate(john), with:

\[
A_1 = \{ \text{candidate(john)} \leftarrow \text{profile_ok(john)}; \text{profile_ok(john)} \leftarrow \text{goodincome(john)}, \text{trustctry(john,krakosia)}; \text{trustctry(john,krakosia)} \leftarrow \text{info(john,krakosia,_,_), credible(krakosia)}; \text{goodincome(john)} \leftarrow \text{info(john,_,_,400), 400>300} \}
\]

Bank criteria as defeasible rules

\[
\begin{align*}
\text{candidate(P)} & \leftarrow \text{profile_ok(P)}. \\
\text{candidate(P)} & \leftarrow \text{info}(P,\_,\_,\text{Income}), \text{req_loan}(P,\text{Amount}), \text{Amount}<\text{Income} \times 10, \text{trustctry}(P,\text{Ctry}). \\
\text{profile_ok(P)} & \leftarrow \text{goodincome}(P), \text{trustctry}(P,\text{Ctry}). \\
\text{trustctry}(P,\text{Ctry}) & \leftarrow \text{info}(P,\text{Ctry},\_,\_), \text{credible}(\text{Ctry}). \\
\text{goodincome}(P) & \leftarrow \text{info}(P,\_,\_,\text{Inc}), \text{Inc}>300. \\
\sim\text{goodincome}(P) & \leftarrow \sim\text{solvent}(P). \\
\sim\text{solvent}(P) & \leftarrow \text{info}(\_,\_,\text{phdstudent,}_). \\
\text{solvent}(P) & \leftarrow \text{info}(\_,\_,\text{phdstudent,}_), \text{richfamily}(P). \\
\text{richfamily}(P) & \leftarrow \text{family_record}(P,\text{rich}).
\end{align*}
\]
Is John a good candidate?

But… there is an argument \( A_2 \) for \(-\text{goodincome}(\text{john})\) that defeats \(<A_1, \text{candidate}(\text{john})>\) with:

\[
A_2 = \{ -\text{goodincome}(\text{john}) \rightarrow \text{solvent} (\text{john});
\text{solvent} (\text{john}) \rightarrow \text{info}(\text{john,_,phdstudent,_,}) \}.
\]

But… there is another argument \( A_3 \) for \( \text{solvent}(\text{john}) \), which defeats on its turn \(<A_2, -\text{goodincome}(\text{john})>\) :

\[
A_3 = \{ \text{solvent}(\text{john}) \rightarrow \text{info}(\text{john,_,phdstudent,_,}), \text{richfamily}(\text{john}) \}
\]

\[<A_1, \text{candidate}(\text{john})> \text{ U }<A_2, -\text{goodincome}(\text{john})> \text{ D }<A_3, \text{solvent}(\text{john})> \text{ U}\]

Therefore john can be justified as a good candidate.

From HTML to XForms...

- **Form**: central structural abstraction for data collection, storage & retrieval in information management systems
- Several programming technologies have been developed which outperformed static Web pages.
- Sophisticated standards have emerged for web-based forms, such as XForms (Dubinko et al, 2003).
- **Proposal**: extend traditional approaches by including defeasible reasoning capabilities encoded in a scripting language based on DeLP, called XDeLP.
- To do this, we will provide a rather generic definition that captures the notions of form schema and form instance, useful for presenting our approach.

Example (revisited)

- A Bank wants to provide a web-based service for requesting loans.
- Potential candidates have to fill in a web-based form in order to request a loan.
- How to reason about candidate profiles on the basis of the bank’s knowledge about them?

**Goal**: to model and reason about features (defeasible attributes) which depend on incomplete and potentially inconsistent information.

Form Schema. Form instance

- A **form schema** is a 2-uple \( F=<F,T> \), where:
  - \( F = \{ f_1, f_2, \ldots, f_n \} \) is a list of **form fields**, and
  - \( T = \{ T_1, T_2, \ldots, T_n \} \) is a list of **types**.

- A **form instance** \( F_V \) based on \( F \) with value \( V \) is a 2-uple \( F_V=<F,V> \) where \( V = \{ v_1,v_2,\ldots,v_n \} \) is list of **values**.

Example

- \( F = \) a form schema:
  - \( F = \) [ name, profession, income, amountreq, country]
  - \( T = \) [ string, string, real, integer, string]
  - \( F_V =<F,V> \) is a form instance:
    - \( V = \) [ john, phdstudent, 400, 2000, krakosia]
Integrating forms with DeLP

- Let $\text{F} = \langle \text{F}, \text{T} \rangle$ be a form schema with $\text{F} = \{ f_1, \ldots, f_n \}$ and $\text{F}_v$ a form instance.
- The set of emerging facts is defined as $\text{facts}(\text{F}_v) = \{ f_1(\text{F}, v_1), f_2(\text{F}, v_2), \ldots, f_n(\text{F}, v_n) \}$
- E.g: Considering the previous form:
  \[ \text{facts}(\text{F}_v) = \{ \text{name}(\text{F}, \text{john}), \text{profession}(\text{F}, \text{phdstudent}), \text{income}(\text{F}, 400), \text{amountreq}(\text{F}, 2000), \text{country}(\text{F}, \text{krakosia}) \} \]

Basic idea: overview

Form Schema $\text{F}$

Delp Program $\text{P}$

D-Forms & DeLP Scripts in XML

Javascript code: example

Programmers usually validate form data by attaching some imperative Javascript code to buttons…

Approach: extending the JS language with primitives for invoking the DeLP engine, and analyzing possible outcomes.

XML Syntax for XDeLP

Note: The text contains mathematical expressions and logical operators, which are not rendered in the image.
Redefining DeLP programs

- Let \( P_1, P_2 \) be two DeLP programs, such that \( P_1 \) defines predicates \( R_1, \ldots, R_m \), and \( P_2 \) defines predicates \( S_1, \ldots, S_n \).
- The redefinition of \( P_1 \) wrt \( P_2 \) is a new program \( P' \):
  \[
  P' = P_1 \otimes P_2 = \{ R_1, \ldots, R_m \} \cup \{ S_1, \ldots, S_n \} \setminus \{ R_i : \exists S_j \in P_2 \text{ with } \text{name}(R_i) = \text{name}(S_j) \land \text{arity}(R_i) = \text{arity}(S_j) \}
  \]

DeLP program \( P_{\text{HSO}} \) from Home Security Office

\[
P_{\text{HSO}} \text{ contains:}
\begin{align*}
\text{country}(\text{greece}, \text{democracy}), \\
\text{country}(\text{krakosia}, \text{democracy}), \\
\text{country}(\text{krakosia}, \text{atwar}), \\
\text{credible}(\text{Ctry}) \rightarrow \text{country}(\text{Ctry}, \text{democracy}), \\
\neg\text{credible}(\text{Ctry}) \rightarrow \text{country}(\text{Ctry}, \text{democracy}), \\
\text{country}(\text{Ctry}, \text{atwar}), \\
\neg\text{credible}(\text{Ctry}) \rightarrow \text{country}(\text{Ctry}, \text{democracy}), \\
\text{country}(\text{Ctry}, \text{comuptigovt}).
\end{align*}
\]

Let \( P' = P_{\text{Bank}} \otimes P_{\text{HSO}} \)

Now the bank can performed a more refined reasoning about its clients by using \( P' \) instead of \( P_{\text{Bank}} \)

Is John a good candidate? (revisited)

- \( \langle A_{\text{I}}, \text{candidate(john)} \rangle^D \)
- \( \langle A_{\text{II}}, \neg\text{goodincome(john)} \rangle^D \)
- \( \langle A_{\text{III}}, \neg\text{credible(krakosia)} \rangle^D \)
- \( \langle A_{\text{IV}}, \text{solvent(john)} \rangle^U \)

Part 2 - Outline

- Introduction to Defeasible Logic Programming (DeLP).
- Application of DeLP: User Support Systems
- Applications of DeLP: Web-Based Forms.
- Extensions of DeLP: P-DeLP, O-DeLP
Classifying Dialectical Trees

Def: Let $T$ be a dialectical tree, such that for every $T'$ evolving from $T$ (i.e., $T \subseteq T'$) it holds that $\text{Mark}(T) = \text{Mark}(T')$.

Then $T$ is said to be a settled dialectical tree.

Def: $T$ is a minimally settled dialectical tree iff there is no $T' \subseteq T$ such that $T'$ is a settled dialectical tree.

Def: $T$ is an optimally settled dialectical tree iff it is minimally settled and there for any other minimally settled tree $T'$, it holds that $\text{cost}(T) \leq \text{cost}(T')$.

Search Space for Dialectical Trees

DeLP : extensions

Recently extensions of DeLP have been developed:

- P-DeLP (Chesñevar et. al, 2004): aims at modelling reasoning under uncertainty (e.g. possibilistic reasoning).

- O-DeLP (Capobianco et. al, 2004): aims at modelling reasoning for agents in changing environments.

Computing Dialectical Trees

If dialectical trees are computed depth-first as AND-OR trees, the order in which branches are generated determines the size of the search space.

DeLP : extensions

DeLP : extensions

DeLP : extensions

Current P-DeLP framework

References:


- [Alsinet-Godo, 2000] : PGL, a possibilistic logic based on Gödel fuzzy logic for reasoning under possibilistic uncertainty and representing vague knowledge.

Why P-DeLP ?

- Problem in MAS: formalizing an agent's knowledge to perform defeasible inferences efficiently. Dealing with uncertainty and fuzziness are common requirements in agents, but not embedded at object level in most argument-based formalisms!

- Proposal: P-DeLP is a LP language that integrates features from argumentation theory and logic programming, incorporating possibilistic uncertainty and fuzzy knowledge at object-language level.

- Ultimate goal: to have an argument-based inference engine (suitable for real-world applications) using P-DeLP, incorporating fuzziness and uncertainty.
PGL: fundamentals

- Underlying logic: PGL, a possibilistic logic based on Gödel fuzzy logic [Alsinet-Godo, 2000].

Syntax

- A set of fuzzy propositional variables \{p, q, …\}
- Connectives: \(\land\) (and), \(\lor\) (or), \(\neg\) (negation)
- Literals: (fuzzy) atom \(q\), or negated (fuzzy) atom \(\neg q\)
- Wff: \(Q \land L_1, \ldots L_n\), where \(Q, L_1, \ldots L_n\) are literals, \(n \geq 0\)
- Certain-weighed clauses: \((\phi, \alpha)\), where \(\phi\) is a wff and \(\alpha \in [0, 1]\) expresses a lower bound for the certainty of \(\phi\) in terms of a necessity measure.

PGL Proof Theory (basics)

- Triviality axiom: \((\phi, 0)\)
- Generalized MP: \([L_1, \beta_1], \ldots, (L_k, \beta_k)\) \((Q \leftarrow L_1, \ldots, L_k, \gamma) (Q, \min(\gamma, \beta_1, \ldots, \beta_k))\)

P-DeLP

Contradictory knowledge: \(\Gamma \vdash \bot\) iff \(\Gamma \vdash (q, \alpha)\) and \(\Gamma \vdash \neg (q, \beta)\), \(\alpha, \beta > 0\)

P-DeLP Program \(P = (\Pi, \Delta)\), where \(\Pi\) is a non-contradictory finite set of certain clauses (\(\alpha = 1\)), and \(\Delta\) is a finite set of uncertain clauses (\(\alpha \leq 1\)).

Argument \((A, Q, \alpha)\): \(A \subseteq \Delta\) is an argument for \(Q\) with necessity measure \(\alpha > 0\) iff \(A, Q, \alpha\)

1. \(A \cup A \vdash (Q, \alpha)\)
2. \(A \cup A\) is not-contradictory
3. \(A\) is minimal wrt. set inclusion

Argument \((S, R, \beta)\) is a subargument of \((A, Q, \alpha)\) iff \(S \subseteq A\)

P-DeLP in an agent’s reasoning module

Sample rules:

- When there is pump clog, fuel is not ok:
  \((\neg fuel\_ok \leftarrow pump\_clog, 1)\)
- When there is heat, usually engine is not ok.
  \((\neg engine\_ok \leftarrow heat, 0.95)\)

Engine has 3 switches on:
- There is heat
- Is the engine ok?

Agent Reasoning Module

- Facts
- Rules
- P-DeLP program
- Dialectical Base
- Inference engine
- Updating mechanism
- Environment
- Sensor input (perception)
- Other Agent (e.g. supervisor agent)
- Queries
- Answers

Query: engine ok?
Answer: No (0.3)
ODeLP-based agent architecture

P-DeLP and O-DeLP: references


END OF PART 2