Logic Programming in a Legal Context

Bob Kowalski, Imperial College London
http://www.doc.ic.ac.uk/~rak/
Introduction to Logic Programming in a Legal Context (~ 15 minutes)

Michael Genesereth, Stanford University
http://logic.stanford.edu/people/genesereth/genesereth.html
Computational Law
The Cop in the Back Seat (~ 20 minutes)

Markus Triska, Austrian Federal Ministry for Digital and Economic Affairs
https://www.metalevel.at/
Logic Programming in modern e-Government Services (~ 20 minutes)

Miguel Calejo, Interprolog.com
http://www.calejo.com/home
Logic Programming and Smart Contracts (~ 20 minutes)

General discussion (~ 15 minutes)

Logic Programming in a Legal Context

Michael Genesereth, Stanford University
http://logic.stanford.edu/people/genesereth/genesereth.html

Computational Law
The Cop in the Back Seat (~ 20 minutes)

Director of the Logic Group at Stanford
Founder and Research Director of CodeX - the Stanford Center for Legal Informatics.
One of the Founders of Teknowledge, CommerceNet, Mergent Systems, SIPX and Symbium.

Launched an effort in 2016 to bring logic education to high schools across America, utilising the same course material as a MOOC on teaching logic.

Books
Genesereth, M.R. and Nilsson, N.J., 2012. Logical foundations of artificial intelligence.
Genesereth, M.R., 2010. Data integration: The relational logic approach.
Genesereth, M.R. and Kao, E., 2013. Introduction to logic. .
Genesereth, M. and Thielscher, M., 2014. General game playing.
Genesereth, M.R. and Chaudhri, V., 2020. Introduction to Logic Programming.

Logic Programming in a Legal Context

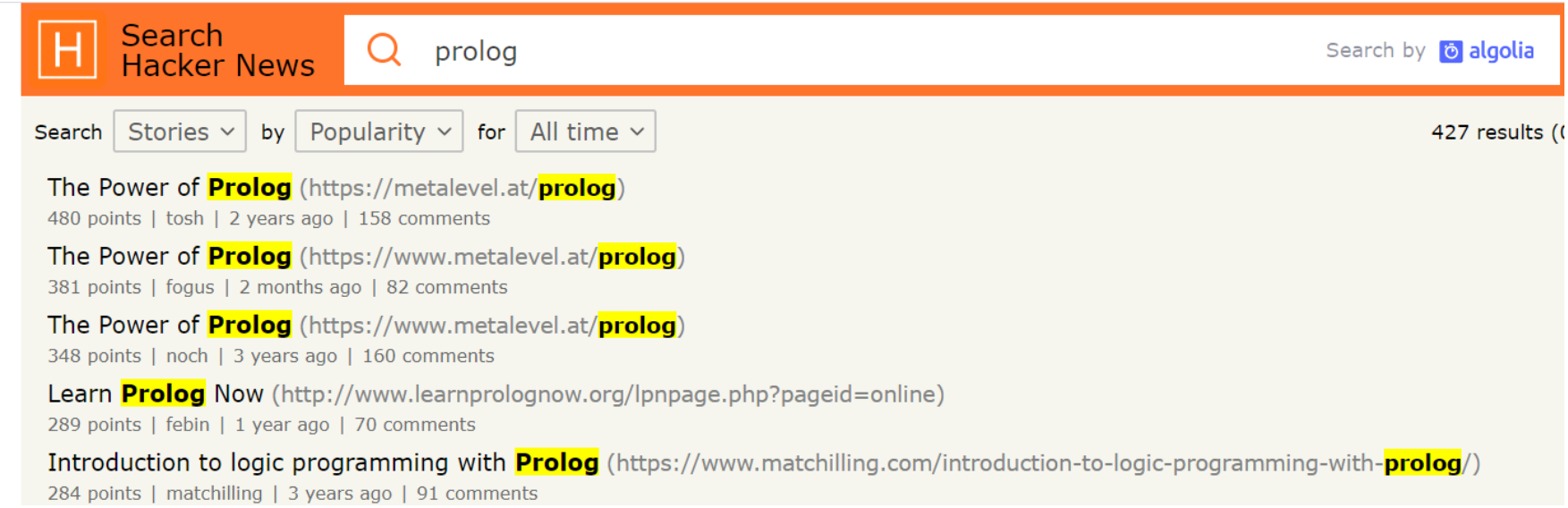
Markus Triska, Austrian Federal Ministry for Digital and Economic Affairs
https://www.metalevel.at/

Logic Programming in modern e-Government Services (~ 20 minutes)

PhD, 2014, Technische Universität Wien
Correctness Considerations in CLP(FD) Systems

SWI-Prolog Jan Wielemaker, Tom Schrijvers, Markus Triska, Torbjörn Lager, TPLP 12 (2012),

Book: The Power of Prolog



Logic Programming in a Legal Context

Miguel Calejo, Interprolog.com
http://www.calejo.com/home

Logic Programming and Smart Contracts (~ 20 minutes)

PhD, 1992, New University of Lisbon
Declarative logic program debugging.

Founded Servisoft, Declarativa, BookMARC, and Renting Point.
One of the Founders of Coherent Knowledge Systems.

Academic Associations with New University of Lisbon,
Universidade Portucalense and Universidade do Minho.

Past President of the Portuguese AI Association, 2012-2015 (appia.pt).
Member of the ISO/IEC JTC1/SC22/WG17 (Prolog standardization) group.

Co-Developer with Bob Kowalski and Fariba Sadri of LPS (Logic Production System)
Wielemaker J, Riguzzi F, Kowalski RA, Lager T, Sadri F, Calejo M., Using SWISH to realize interactive web-based tutorials for logic-based languages. TPLP 2019.

Logic Programming in a Legal Context

Alternative approaches to the representation and execution of legal texts:

- Conditional “logic” in imperative programming languages
- Condition-action rules in expert system languages
- Deontic logics of obligation, prohibition and permission
- Defeasible logics of rules and exceptions
- Functional programming
- Logic programming

Logic programming

From Wikipedia, the free encyclopedia

"Rule-Based" redirects here. For the method of machine translation, see Rule-based machine translation. For methods of machine learning, see Rule-based machine learning.

Logic programming is a programming paradigm which is

Programming paradigms

Logic programming rules are commonly confused with

Reactive rules:
Condition-action rules
Event-condition-action rules
Active rules

Logic programming is a programming paradigm which is largely based on formal logic. Any program written in a logic programming language is a set of sentences in logical form, expressing facts and rules about some problem domain. Major logic programming language families include Prolog, answer set programming (ASP) and Datalog. In all of these languages, rules are written in the form of *clauses*:

H :- B_1, …, B_n.

and are read declaratively as logical implications:

H if B_1 and … and B_n.

Conditional “logic” in imperative programming languages and condition-action rules in expert system languages

Have the conditional imperative form:

If conditions
Then do actions.

Do not have an accepted logical semantics.

Defeasible logics of rules and exceptions

Rule: conclusion **if** conditions.

Exception: **the rule does not apply unless** restrictions.

can be translated into the simpler logic programming form:

Rule with exception:

conclusion **if** conditions **and** restrictions.

Deontic logics of obligation, prohibition and permission

If conditions **then** fulfil obligations.
If conditions **and not** fulfil obligations **then** penalty.

In logic programming, the second sentence is sufficient:

penalty **if** conditions **and not** fulfil obligations.

Functional programming for smart contracts

Accord Ergo mostly written in Coq
Tezos implemented in OCaml
Cardano blockchain written in Haskell
Æternity written in Erlang

Logic programs can be
nearly isomorphic to natural language legal texts, and
therefore nearly “self-verifying”.

General discussion

If LP is so well suited for legal applications,
why aren't there more of them?

Can LP representations of legal applications really be self-verifying?

What do functional programmers mean when they say functional programs facilitate specification and verification?

Are LP implementations not yet good enough for legal applications?

Relationship with Legal RuleML?

Is a CNL (Controlled Natural Language) syntactic sugar for LP, the answer to all of our problems?

Is translation from uncontrolled NL to LP possible and/or desirable?