The Production Logic Programs Approach: KR Foundations for Semantic Rules on the Web

Prof. Benjamin Grosof

MIT Sloan School of Management

Information Technologies group

http://ebusiness.mit.edu/bgrosof

Presentation (1-hour) at

Semantic Web Seminar,

Stanford University Computer Science Dept.

Stanford, CA, USA, Mar. 29, 2006

Hosted by Prof. Michael Genesereth and Michael Kassoff

Quickie Bio of Presenter Benjamin Grosof

- MIT Sloan professor since 2000
- 12 years at IBM T.J. Watson Research; 2 years at startups
- PhD Comp Sci, Stanford;   BA Applied Math Econ/Mgmt, Harvard
- Semantic web services is main research area:
  - Rules as core technology
  - Business Applications, Implications, Strategy:
    - e-contracting/supply-chain;   finance; trust; …
  - Overall knowledge representation, e-commerce, intelligent agents

- Co-Founder, Rule Markup Language Initiative – the leading emerging standards body in semantic web rules (http://www.ruleml.org)
- Area Editor, Semantic Web Services Initiative – which coordinates world-wide SWS research and early standards (http://www.swsi.org)

Resources for More Info

- On author's website (http://ebusiness.mit.edu/bgrosof), see especially:
  1. ISWC-2005 Rules Tutorial slideset (half-day conference tutorial, 200+ detailed slides)
     - …/#ISWC2005RulesTutorial
     - "Semantic Web Rules with Ontologies, and their E-Service Applications"
       - A. Core technology: knowledge representation languages, theory, and techniques; standards design
       - B. Tools: rule inference engines, translators between rule systems/languages, ontology integration
       - C. Applications in E-Services: semantic mediation and ontology translation, e-contracting, trust policies, financial reporting
         - Business value analysis, market roadmapping

  2. Production Logic Programs paper & info
     - …/#ProductionLogicPrograms (or just …/#PLP)

Also:

- Recent talks (including this one soon), not just papers
- SweetRules toolset (http://sweetrules.projects.semwebcentral.org)

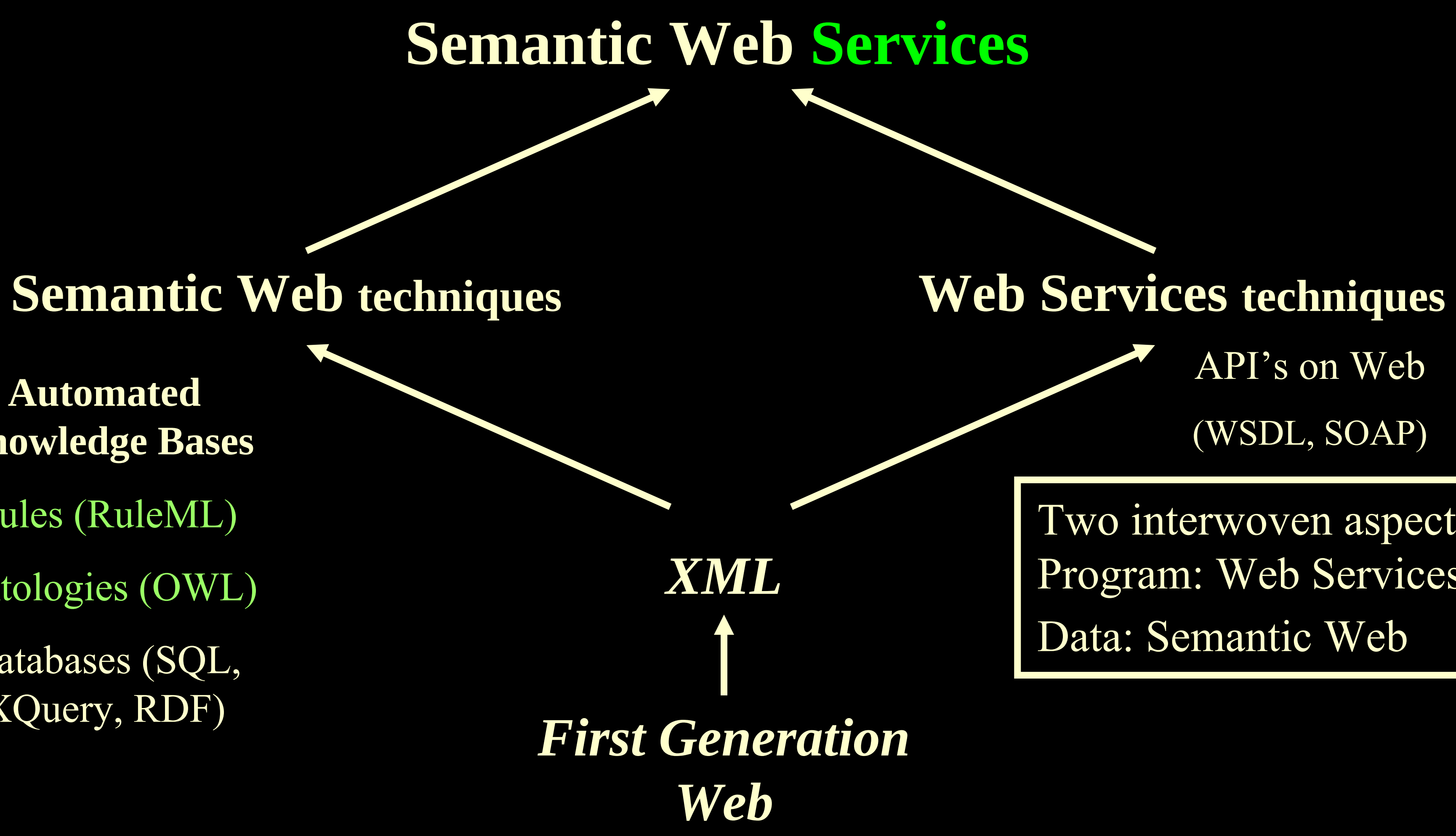
Outline

- Introduction
  - Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

Talk Mode: the MIT Firehose

Shortened from a 90-minute talk
$\Rightarrow$ Some skimmed

Next Generation Web



Automated Knowledge Bases

Rules (RuleML)

Ontologies (OWL)

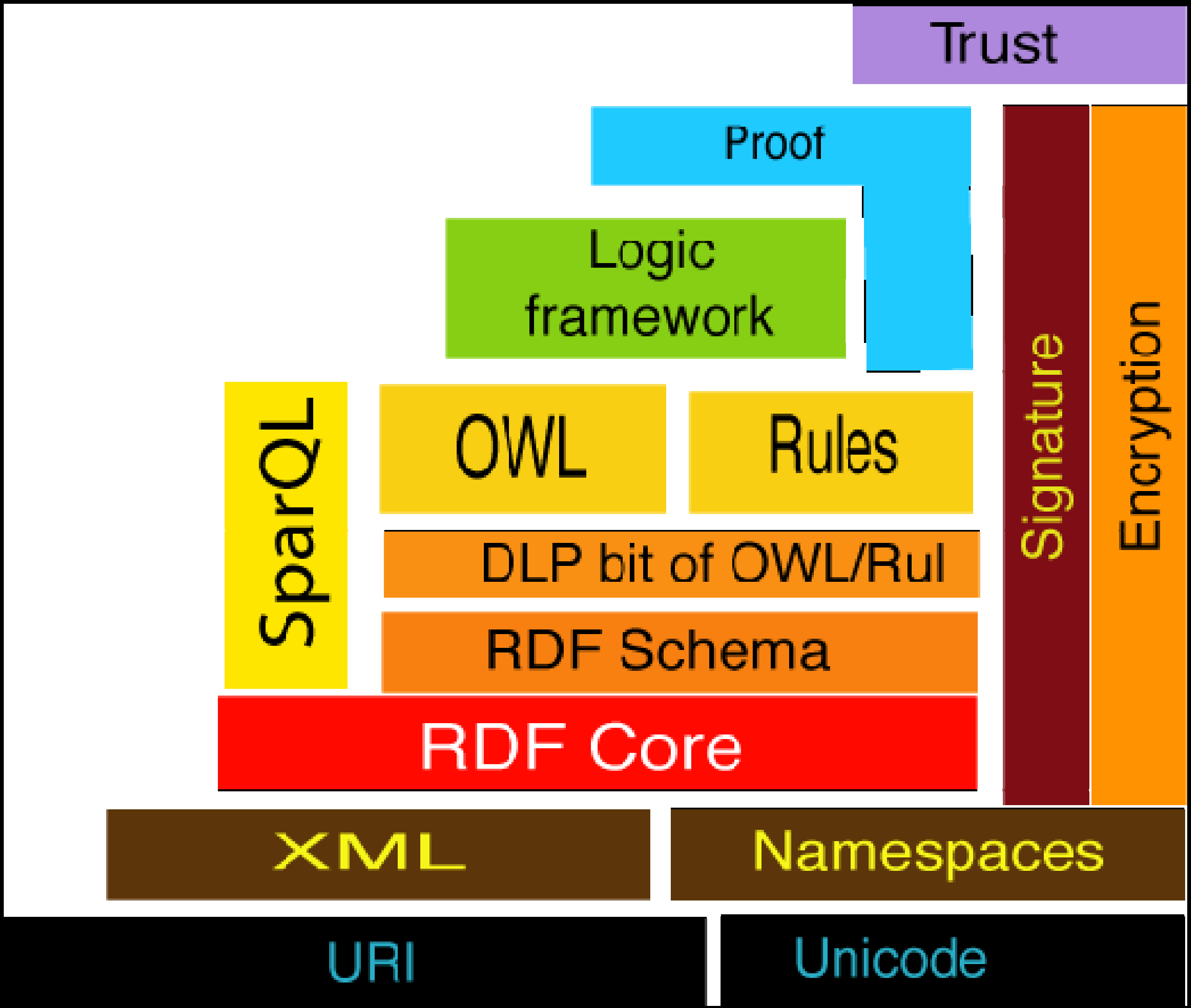
Databases (SQL, XQuery, RDF)

API's on Web

(WSDL, SOAP)

Two interwoven aspects:
Program: Web Services
Data: Semantic Web

2005 W3C Semantic Web "Stack": Standardization Steps



Semantic Web Services

- Convergence of Semantic Web and Web Services
- Consensus definition and conceptualization still forming
- Semantic (Web Services):
  - Knowledge-based service descriptions, deals
    - Discovery/search, invocation, negotiation, selection, composition, execution, monitoring, verification
    - <u>Advantage: **reuse** of knowledge across app's, these tasks</u>
  - Integrated knowledge
- (Semantic Web) Services: e.g., infrastructural
  - Knowledge/info/DB integration
  - Inferencing and translation

Flavors/Families of Rules Commercially Most Important today in E-Business

- E.g., in OO app's, DB's, workflows.
- "CCI" = Currently Commercially (most) Important

1. Relational databases (RDBMS), SQL: Views, queries, facts are all rules.
    - XQuery, SPARQL emerging. SQL99 even has recursive rules.
2. Production rules (OPS5 and CLIPS heritage): e.g.,
    - Fair Isaac, ILOG, Haley, etc.
3. Event-Condition-Action rules (loose family similar to PR), cf.:
    - business process automation / workflow tools.
    - active databases; publish-subscribe
4. Prolog. *"logic programs" as a full programming language.*
    - *"Pure" Prolog – declarative LP subset, has no cuts or external procedure calls, does backward inferencing in declarative LP*
5. *(Lesser: other knowledge-based systems, and things hard to classify or further from declarative such as some "business rule" systems.)*

Production Rules (PR): History

- Grand-daddy: OPS5 research system at CMU in ’70’s
- NOT declarative
- CLIPS *system*: open PR system done by US govt ~ a decade ago
- CLIPS *syntax*: used with tweaks by many current PR systems
- Have incremental/dynamic/updating capabilities
- PR reengineered in ’90’s to be fine-grain embedded in C++/Java etc. programming language, with access to those external objects
- OMG PRR standards effort since late 2003
- *Full PR systems often also have scripting and a kind of backward-inferencing capability; for semantically interoperable web rules that has not been the focus (at least initially).*

Event-Condition-Action Rules (ECA): History

- Loose family; no consensus/standard detailed formulation/abstraction
- Fairly similar to Production Rules: forward, Conditions, Actions
  - NOT declarative
- Plus there's the "Event" notion (see next slide)
- More focus than PR on incremental inferencing and specialized optimizations around "Events"

- Many database systems have ECA capabilites, e.g., for transactional triggers or pub-sub.
- In '90's became used widely for loose coupled business process automation / workflow / integration / orchestration

Events in ECA Rules

- "Event" is a kind of premise info update, <u>and</u> a kind of control trigger for incremental inferencing
  - This conflates declarative and procedural aspects! *Challenge ;-)*
  - Often not precisely described, for given ECA system/language
- "Event" part of a rule body is a kind of condition, and control "port"
- Often there's "complex event processing" with specialized treatment
  - E.g., event sublanguage and special processors for generating/testing events
- *History* of event updates/info-states is often important

- Other flavors of rules can also do events and incremental inferencing, to varying degrees

Our Research Aspects/Questions about the Semantic Web

- Core technologies: Requirements, concepts, theory, algorithms, standards?
  - Rules in combination with ontologies; probabilistic, decision-/game-theoretic

- Business applications and implications: concepts, requirements analysis, techniques, scenarios, prototypes; strategies, business models, market-level evolution?
  - End-to-end e-contracting, finance, trust; …

Some Answers to: “Why does SW Matter to Business?”

- 1. “Death. Taxes. Integration.” - They’re always with us.

- 2. “Business processes require communication between organizations / applications.” - Data and programs cross org./app. boundaries, both intra- and inter- enterprise.

- 3. “It’s the *automated knowledge* economy, stupid!” - The world is moving towards a knowledge economy. And it’s moving towards deeper and broader automation of business processes. The first step is automating the use of <u>structured</u> knowledge.
  – Theme: *reuse* of knowledge across multiple tasks/app’s/org’s

Strategic Business Foci in our SW Research

- Knowledge-based Services Engineering: intra- and inter- enterprise
- Target "killer app" known for 30 years: do better job of EDI
- Challenges:
  - Ease of development, deployment ↑
  - Reuse of knowledge ↑
  - ⇒ life cycle costs ↓ , agility ↑
- Starting with: Policies
  - Using recent theory breakthroughs in semantic rules
  - E.g., for end-to-end contracting and authorization (incl. security)
- Starting with: EAI as well as B2B

SWS and Rules   Summary

*** SWS Tasks Form 2 Distinct Clusters,*

each with associated Central Kind of Service-description Knowledge and Main KR

1. Security/Trust, Monitoring, Contracts, Advertising/Discovery, Ontology-mapping Mediation
   - Central Kind of Knowledge: Policies
   - Main KR: Nonmon LP (rules + ontologies)
2. Composition, Verification, Enactment
   - Central Kind of Knowledge: Process Models
   - Main KR: FOL (axioms + ontologies)
     - + Nonmon LP for ramifications (e.g., cf. Golog)

- Thus RuleML & SWSF specify both Rules, FOL
  - Fundamental KR Challenge: "Bridging" Nonmon LP with FOL
    - SWSF experimental approach based on hypermon. [Grosof & Martin]

SW Rules: Use Cases from our research

- Contracts/negotiation, advertising/discovery
  - E-procurement, E-selling
  - Pricing, terms & conditions, supplier qualification, …

- Monitoring:
  - Exception handling, e.g., of contract violations
    - Late delivery, refunds, cancellation, notifications
  - Notifications, personal messaging, and other workflow

- Trust Policies: authorization, confidentiality & privacy, security, access control
  - E.g., financial services, health care
    - *Extensive analysis of business case/value*

- Semantic mediation: rule-based ontology translation, context-based information integration

Semantic Rules News

News recently:

- Fundamental theory and technique breakthroughs, e.g.:
  - Declarative logic programs (LP) basis for interoperability, then webized → RuleML standards design (2001-)
  - Courteous LP prioritized defaults, robust modular merging
  - Description LP ontology integration
  - Production LP interoperability+semantics for production rules, declarative procedural attachments for actions and queries
  - SweetRules V2 open source toolset platform (2004-)
- Large US, EU research projects (DAML, WSMO) focus on rules (DARPA Agent Markup Language; Web Service Mediation Ontology)
- RuleML standards design gets large mindshare for its technical approach

Semantic Rules News (cont.'d)

News recently:

- W3C forms Rule Interchange Format WG, full standards effort, after holding a Workshop (Dec. 2005)
- OMG forms standards efforts on production rules, rule management
- Semantic Web Services Framework design (2005) focuses on rules
- Rule-based Policy area heats up in web services, semantic web, incl. at Oasis.   Oasis forms Semantic Execution Env. standards effort (2005).
- Semantic web rules workshop series becomes full research conference (RuleML-2005)   colocated with ISWC
- Gartner etc. reports on rules sector

Advantages of Standardized SW Rules for Policies, e.g., Authorization/Security

- Easier Integration: with rest of business policies and applications, business partners, mergers & acquisitions
  - Enterprise integration, B2B
- Familiarity, training
- Easier to understand and modify by humansChange management
- Quality and Transparency of implementation in enforcement
  - Provable guarantees of behavior of implementation
- Reduced Vendor Lock-in
- Expressive power
  - Principled handling of conflict, negation, priorities
- $\Rightarrow$ Agility, change management $\uparrow$

Advantages of SW Rules, cont'd:
Loci of Business Value in Policy Management

- Reduced system dev./maint./training costs
- Better/faster/cheaper policy admin.
- Interoperability, flexibility and re-use benefits
- Greater visibility into enterprise policy implementation ⇒ better compliance
- Centralized ownership and improved governance by Senior Management
- Rich, expressive policy management language allows better conflict handling in policy-driven decisions
- Strategic agility, incl. wrt business model

Outline

- Introduction
  - Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

Semantic Rules: Differences from Rules in the 1980's / Expert Systems Era

- Get the KR right (knowledge representation)
  - More mature research understanding
  - Semantics independent of algorithm/implementation
  - Cleaner; avoid general programming/scripting language capabilities
  - Highly scaleable performance; better algorithms; choice from interoperability
  - Highly modular wrt updating; use prioritization
  - → Highly dynamic, scaleable rulebase authoring: distributed, integration, partnering
- Leverage Web, esp. XML
  - Interoperable syntax
  - Merge knowledge bases
- Embeddable
  - Into mainstream software development environments (Java, C++, C#); not its own programming language/system (cf. Prolog)
- Knowledge Sharing: intra- or inter- enterprise
- Broader set of Applications

New Fundamental Rule KR Theory

that enables Key Technical Requirements for SWS

In 1985-94:

- Prolog interoperable with relational DB; LP extends core-SQL [many]
- Richer logical connectives, quantifiers [Lloyd & Topor]
- "Well Founded" Semantics for Negation-As-Failure [Van Gelder et al; Przmusinski]
- Hilog quasi-higher order expressiveness, meta-syntax flexibility [Kifer et al.]
- Frame syntax cf. F-Logic [Kifer *et al.*]

In 1995-2004:

- Courteous LP: prioritized conflict handling [Grosof]
  - Robust, tractable, modular merging & updating
- Situated LP: hook rules up to services [Grosof]
- Description LP: combine Description Logic ontologies [Grosof *et al.*]
- Courteous Inheritance: combine OO default ontologies [Grosof *et al.*]
- Production Rules as LP: interoperate [Grosof *et al.*]
  - Declarative LP as interoperable core between commercial families [Grosof *et al.*]
- Hypermonotonic Reasoning: combine with FOL [Grosof (in-progress)]

Concept of Knowledge Representation (KR)

- A knowledge representation S is defined as a triple (LP, LC, |=), where:
  - LP is a formal language of sets of premises (i.e., premise expressions)
  - LC is a formal language of sets of conclusions (i.e., conclusion expressions)
  - |= is the entailment relation.
    - Conc(P,S) stands for the set of conclusions that are entailed in KR S by a set of premises P
    - We assume here that |= is a functional relation.

Example of Entailment: Mortality

- In First-Order Logic (FOL) KR:
  - Let P be the premises:
  - $\forall$?X. human(?X) $\Rightarrow$ mortal(?X).
  - human(Socrates).
  -
  - In FOL, P entails (among others) the conclusion:
    - mortal(Socrates).
  - Notation:
    - “$\forall$” means “for all”.
    - “?” Prefixes a logical variable.

Example of Entailment: Discounting

- In the Courteous Logic Programs KR (e.g., RuleML):

  Let P be the premises:

  - {loyald} discount(?cust, RamadaHotel, 10percent)
    $\leftarrow$ memberOf(?cust, AAA).
  - {seniord} discount(?cust, RamadaHotel, 25percent)
    $\leftarrow$ age(?cust, ?x) and greaterThan(?x, 64).
  - overrides(seniord, loyald).
  - $\perp \leftarrow$ discount(?c, ?h, ?y) and discount(?c, ?h, ?z) | (?y $\neq$?z).
  - memberOf(Faisal, AAA).
  - age(Faisal, 72).

  - In this KR, P entails (among others) the conclusion:
    discount(Faisal, RamadaHotel, 25percent).

Example of Discounting, cont.'d

In the more general Production Logic Programs KR:

Suppose one adds the rule:

– @emailCouponAd(?cust, RamadaHotel, ?x)
  ← discount(?cust, RamadaHotel, ?x).

Then P entails the action (i.e., sanctions a call to an attached procedure):

@emailCouponAd(Faisal, RamadaHotel, 25percent).

KR: What's the Game? Desiderata

- Expressiveness: what can be said
  - useful, natural, complex enough
- Syntax: encoding data format -- e.g., in XML
  - easy enough to edit and communicate, by computers and by humans
- Semantics: principles of sanctioned inference, independent of reasoning algorithms:
  - clear, useful, natural, and understandable enough
- Computational Tractability (esp. worst-case): scale up in a manner qualitatively similar to relational databases: computation cycles go up as a polynomial function of input size
- Reasoning algorithms (compute the entailed conclusions):
  - sound (correct), complete, efficient, clear, and simple enough to engineer

History of Declarative LP

- Developed as a theoretical abstraction of RDBMS and pure Prolog
- Negation well understood by ’94 (not well understood before ’84)
- A number of expressive extensions in the last two decades
- Lots of algorithmic insights, expertise, good implementations
  - E.g., XSB; Flora-2, SweetRules on top of XSB
- For function-free case, similar tractability to RDBMS/SQL. (vs. FOL intractable.)
- Cannot do “reasoning by cases”, i.e., draw disjunctive conclusions and then chain on them in branched fashion. This is essential to the attractive computational complexity. E.g., not good for general constraint solving *a la* complex scheduling.
- Conclusions are essentially (reducible to) ground literals.

Declarative Logic Programs KR

- Basic case: Definite Horn function-free equality-free LP. Tractable. Same as core SQL but with no limitation to backward direction of inferencing.
- A number of extensions -- and restrictions …
  - thus an extensible family forming a lattice.
  - "LP" can mean the family or a member of it
  - "foo LP" can mean a sublattice or member of it
- Datatypes
- Logical functions. Loses tractability.
- Default negation (scoped), a.k.a. Negation-As-Failure

Declarative Logic Programs KR II

- Courteous LP: prioritized conflict handling; strong negation too. Defeasible Logic similar.
  - Guarantees consistent set of conclusions; wrt mutex's (mutual exclusion integrity constraints)
  - Reducible to (default) negation
- Production LP (generalizes Situated LP): procedural attachments for side-effectful actions and tests/queries.
  - Enables interoperability with Production Rules and similar ECA.
  - Built-ins as simple case: for arithmetic, string, etc. comparisons/operations
- Frame syntax cf. F-Logic
- Hilog cf. F-Logic: "meta-linguistic"/"reflection" capability

Declarative Logic Programs KR III

- Lloyd-Topor enhanced logical connectives and quantifiers
  - Reducible to (default) negation
- Skolemization, e.g., for existentials, RDF blank-nodes
- Reification: kind of quotation: belief formula becomes logical term
- Explicit equality: in heads of rules; with background special axioms
- Integrity constraints: reporting vs. inconsistency-generating

- A few other things too

- See SWSL report for a fairly good overview (it omits procedural attachments, however!)

Outline

- Introduction
  - Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

Production Logic Programs KR Extension / Feature: Overview

- Extension Relative to Horn LP
  - Combines orthogonally with LP extensions for default negation (Normal), Courteous, Lloyd-Topor, Frame syntax, Hilog, and several other LP extensions.
- Introduces test and action expressions into the language.
- Declarative approach to procedural attachments.
  - Reformulates our previous Situated extension of LP.
    - That has separate sensor and effector statements that associate an aproc with a predicate.
- Captures heart of production rules (PR) and event-condition-action rules (ECA).
  - We've used PLP KR to provide the first declarative semantics for PR (and ECA): a large fragment of their expressiveness.

Production LP Feature Syntax I

- Additional language entity: Attached Procedure (aproc)
- Two kinds of aproc's:
  - Sensor, a.k.a. test
    - E.g., @PhoneNumberOf(?person,?num)
  - Effector, a.k.a. action
    - E.g., @ChargeCreditCard(?cardnum,?payee,?amt,?time)
- Aproc's can appear in lieu of predicates to form *situated* atoms: sensor atoms or effector atoms.
  - Appear in premise rules, queries, or conclusions.
- Sensor atoms (tests) appear only in rule <u>body</u> (or query)
- Effector atoms (actions) appear only in rule <u>head</u> (or conclusion)

Production LP Feature Syntax II

- Additional language statement type:
  - Binding Restriction (br) statement:

    = Pragma that specifies restrictions on binding patterns for a sensor aproc – wrt bindings available when it's called

    - E.g, bindreq @PhoneNumberOf (BOUND,FREE).
  - There may be multiple br statements for the same sensor aproc.
    - E.g., also: bindreq @PhoneNumberOf (FREE,BOUND).
  - Special cases: all-bound, all-free.
  - If no br statements for a sensor, then it's all-bound.
- Concept of whether a PLP is "br-safe", i.e., satisfies all the binding restriction statements.
  - Can be statically checked, tractably.
    - Analyze each rule body wrt bindings supplied by its atoms, starting with the non-sensor atoms.

Production LP Feature Semantics I

- Actions are sanctioned as part of the overall conclusions.
  - More precisely: a (possibly empty) bag of (ground) actions.
    - "situated" entailment/inferencing; "inferencing+action"
- Sensors read a single initial state of the external environment
- A bag of actions specifies a transition to a new state of the external environment
  - Each satisfying instance of a rule body triggers a head action instance

- External here means external to the pure-belief language.
  - Can view as external to an ideal entailment engine and its internal state
- Concept of an *episode* of entailment/inferencing. Can view as:
  - Sensing and pure-belief entailment in the initial state
  - Effecting immediately after that

PLP Feature Semantics -- Comments

- Thus there's *discipline* in the use of attached procedures:
  - Sensors are *side-effect-free.*
  - Effectors are *engine-safe*: do not affect the engine state and behavior, nor the premises.
  - Sensors and effectors are *snapshot*: sequence and time of call do not matter.
  - Side-effect-ful aproc calls do not appear in the body
    - Unlike Prolog, which does permit them and thus has control-strategy-dependent side-effects (in the general, impure case).
- KR stays declarative
  - control-strategy-independent notion of semantics

Example of Entailment: Discounting

- In the Courteous Logic Programs KR (e.g., RuleML):
  Let P be the premises:
  - {loyald} discount(?cust, RamadaHotel, 10percent) $\leftarrow$ memberOf(?cust, AAA).
  - {seniord} discount(?cust, RamadaHotel, 25percent) $\leftarrow$ age(?cust, ?x) and greaterThan(?x, 64).
  - overrides(seniord, loyald).
  - $\perp \leftarrow$ discount(?c, ?h, ?y) and discount(?c, ?h, ?z) | (?y $\neq$?z).
  - memberOf(Faisal, AAA).
  - age(Faisal, 72).

  - In this KR, P entails (among others) the conclusion:
    discount(Faisal, RamadaHotel, 25percent).

Example of Discounting, cont.'d

In the more general Production Logic Programs KR:

Suppose one adds the rule:

– @emailCouponAd(?cust, RamadaHotel, ?x)
  ← discount(?cust, RamadaHotel, ?x).

Then P entails the action (i.e., sanctions a call to an attached procedure):

@emailCouponAd(Faisal, RamadaHotel, 25percent).

Outline

- Introduction
  - Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

Ubiquity of Priorities in Commercially Important Rules -- and Ontologies

- Updating in relational databases
  - more recent fact *overrides* less recent fact
- Static rule ordering in Prolog
  - rule earlier in file *overrides* rule later in file
- Dynamic rule ordering in production rule systems (OPS5)
  - "meta-"rules can specify agenda of rule-firing sequence
- Event-Condition-Action rule systems rule ordering
  - often static or dynamic, in manner above
- Exceptions in default inheritance in object-oriented/frame systems
  - subclass's property value *overrides* superclass's property value, e.g., method redefinitions
- All lack Declarative KR Semantics

Semantical KR Approaches to Prioritized LP

The currently most important for Semantic Web are:

1. Courteous LP
   - KR extension to Ordinary LP
   - In RuleML, since 2001
   - Commercially implemented and applied
     - IBM CommonRules, since 1999
2. Defeasible Logic
   - Closely related to Courteous LP
     - Less general wrt typical patterns of prioritized conflict handling needed in e-business applications
     - In progress: theoretical unification with Courteous LP

Courteous LP: the What

- Updating/merging of rule sets: is crucial, often generates conflict.
- Courteous LP's feature prioritized handling of conflicts.
- Specify scope of conflict via a set of *pairwise* mutual exclusion constraints.
  - E.g., $\perp \leftarrow$ discount(?product,5%) $\wedge$ discount(?product,10%) .
  - E.g., $\perp \leftarrow$ loyalCustomer(?c,?s) $\wedge$ premiereCustomer(?c,?s) .
  - Permit classical-negation of atoms: $\neg$p means p has truth value *false*
    - implicitly, $\perp \leftarrow p \wedge \neg p$ for every atom p.
- **Priorities** between rules: partially-ordered.
  - Represent priorities via reserved predicate that compares rule labels:
    - overrides(rule1,rule2) means rule1 is higher-priority than rule2.
    - Each rule optionally has a rule label whose form is a functional term.
    - overrides can be reasoned about, just like any other predicate.

Priorities are available and useful

- Priority information is naturally available and useful. E.g.,
  - recency: higher priority for more recent updates.
  - specificity: higher priority for more specific cases (e.g., exceptional cases, sub-cases, inheritance).
  - authority: higher priority for more authoritative sources (e.g., legal regulations, organizational imperatives).
  - reliability: higher priority for more reliable sources (e.g., security certificates, via-delegation, assumptions, observational data).
  - closed world: lowest priority for catch-cases.

- Many practical rule systems employ priorities of some kind, often implicit. E.g.,
  - rule sequencing in Prolog and production rules.
    - Courteous LP subsumes this as special case (totally-ordered priorities), plus enables: merging, more flexible & principled treatment.

Courteous LP: Advantages

- Facilitate updating and merging, modularity and locality in specification.
- Expressive: classical negation, mutual exclusions, partially-ordered prioritization, reasoning to infer prioritization.
- Guarantee consistent, unique set of conclusions.
  - **Mutual exclusion is enforced**. E.g., never conclude discount is both 5% and that it is 10%, nor conclude both p and $\neg$p.
- Scaleable & Efficient: low computational overhead beyond ordinary LP's.
  - Tractable given reasonable restrictions (VB Datalog):
    - extra cost is equivalent to increasing v to (v+2) in Ordinary LP, worst-case.
  - By contrast, more expressive prioritized rule representations (e.g., Prioritized Default Logic) add NP-hard overhead.
- Modular software engineering:
  - via courteous compiler: CLP $\rightarrow$ OLP.
    - A radical innovation. Add-on to variety of OLP rule systems. $O(n^3)$.

EECOMS Example of Conflicting Rules: Ordering Lead Time

- Vendor’s rules that prescribe how buyer must place or modify an order:
- A) 14 days ahead if the buyer is a qualified customer.
- B) 30 days ahead if the ordered item is a minor part.
- C) 2 days ahead if the ordered item’s item-type is backlogged at the vendor, the order is a modification to reduce the quantity of the item, and the buyer is a qualified customer.

- Suppose more than one of the above applies to the current order? **Conflict!**

- Helpful Approach: **precedence** between the rules. Often only *partial* order of precedence is justified. E.g., C > A.

Courteous LP's: Ordering Lead Time Example

- <leadTimeRule1> orderModificationNotice(?Order,14days)
- $\leftarrow$ preferredCustomerOf(?Buyer,?Seller) $\wedge$
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule2> orderModificationNotice(?Order,30days)
- $\leftarrow$ minorPart(?Buyer,?Seller,?Order) $\wedge$
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule3> orderModificationNotice(?Order,2days)
- $\leftarrow$ preferredCustomerOf(?Buyer,?Seller) $\wedge$
- orderModificationType(?Order,reduce) $\wedge$
- orderItemIsInBacklog(?Order) $\wedge$
- purchaseOrder(?Order,?Buyer,?Seller) .
- overrides(leadTimeRule3 , leadTimeRule1) .
- ($\perp \leftarrow$ orderModificationNotice(?Order,?X) $\wedge$
- orderModificationNotice(?Order,?Y)) $\leftarrow$ (?X $\neq$?Y) .

Courteous LP Semantics: Prioritized argumentation in an opposition-locale.

Conclusions from opposition-locales previous to this opposition-locale {p1,...,pk}

(Each pi is a ground classical literal. $k \geq 2$.)

↓

$\boxed{\text{Run Rules for p1,...,pk}}$

↓

Set of Candidates for p1,...,pk:
Team for p1, ..., Team for pk

↓

$\boxed{\text{Prioritized Refutation}}$

↓

Set of Unrefuted Candidates for p1,...,pk:
Team for p1, ..., Team for pk

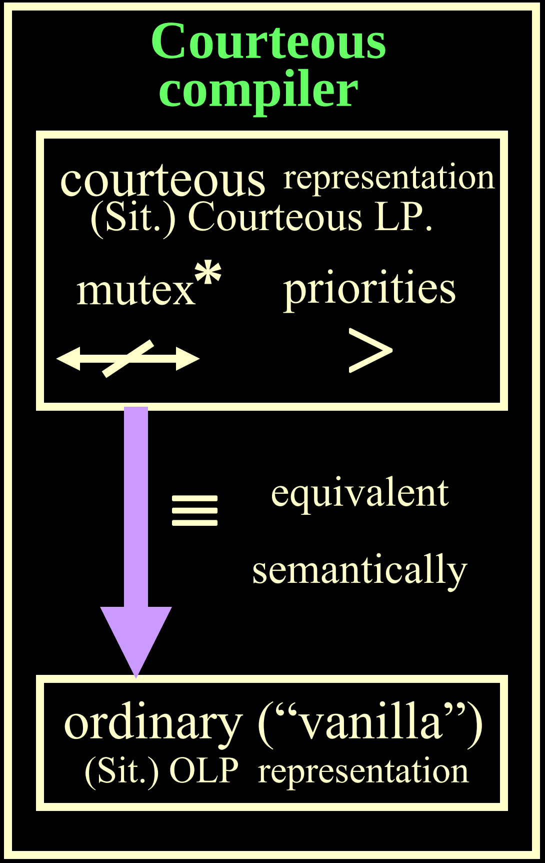
↓

$\boxed{\text{Skepticism}}$

↓

Conclude Winning Side if any: at most one of {p1,...,pk}

Courteous feature: compileable, tractable



Sit. = Situated

* classical negation too

Tractable compilation:

O(n^3), often linear

Tractable inference: e.g., worst-case when no logical functions ("Datalog") & bounded v = |var's per rule| is equivalent to OLP with v $\rightarrow$ (v+2)

Preserves ontology.

Plus extra predicates for

- phases of prioritized argumentation (refutation, skepticism)
- classical negations

Summary: Courteous (Situated) LP's as Core KR

- Key Observations about Declarative OLP:
  - captures common core among commercially important rule systems.
  - is expressive, tractable, familiar.
  - advantages compared to classical logic / ANSI-draft KIF:
    - + + logical non-monotonicity, negation-as-failure.
    - – – disjunctive conclusions.
    - + + tractable.
    - + + procedural attachments: Situated LP's.
- Cleverness of Courteous extension to the OLP representation:
  - prioritized conflict handling $\rightarrow$ modularity in specification. And consistency.
  - courteous compiler $\rightarrow$ modularity in software engineering.
  - mutex's & conflict locales $\rightarrow$ keep tractability. (Compiler is O(n^3).)

Courteous Compiler

- Transformer compiles a courteous LP into an ordinary LP.
- A radically innovative approach in rules representation.
- "Compiles away" conflict, as modular add-on to rule system X's
  - inferencing
  - specification
- Enables courteous features to be added to, or implemented in, a variety of rule systems.
- Tractable: O(n^3). Incremental.

Courteous LP's: Keys to Tractability

- Overall: mutex's & conflict locales $\rightarrow$ keep tractability.
- LP's: disallow disjunctive conclusions, essentially. Classical allows $\Rightarrow$ NP-hard.
- LP's: disallow contraposition (= $\{\neg a \leftarrow .\ ,\ a \leftarrow b \wedge c.\} \Rightarrow (\neg b \vee \neg c)\}$) which requires disjunctive conclusions. "Directional". Classical allows $\Rightarrow$ NP-hard.
- Highly expressive prioritized rule representations (e.g., Prioritized Default Logic, Prioritized Circumscription) allow minimal conflict sets of arbitrary size $\Rightarrow$ NP-hard overhead for conflict handling.
- Courteous conflict handling involves essentially only pairwise conflicts, i.e., minimal conflict sets of size 2. (Current work: possibly generalize to size k.)
  - Novelty: generalize to pairwise mutex's beyond $\perp \leftarrow p \wedge \neg p$, e.g., partial-functional, thus avoid need for contraposition and larger conflict sets.
- Courteous conflict handling is local within an opposition locale: a set of rules whose heads oppose each other through mutex's. Refutation and Skepticism are applied within each locale.

Outline

- Introduction
  - – Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - – Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - – FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

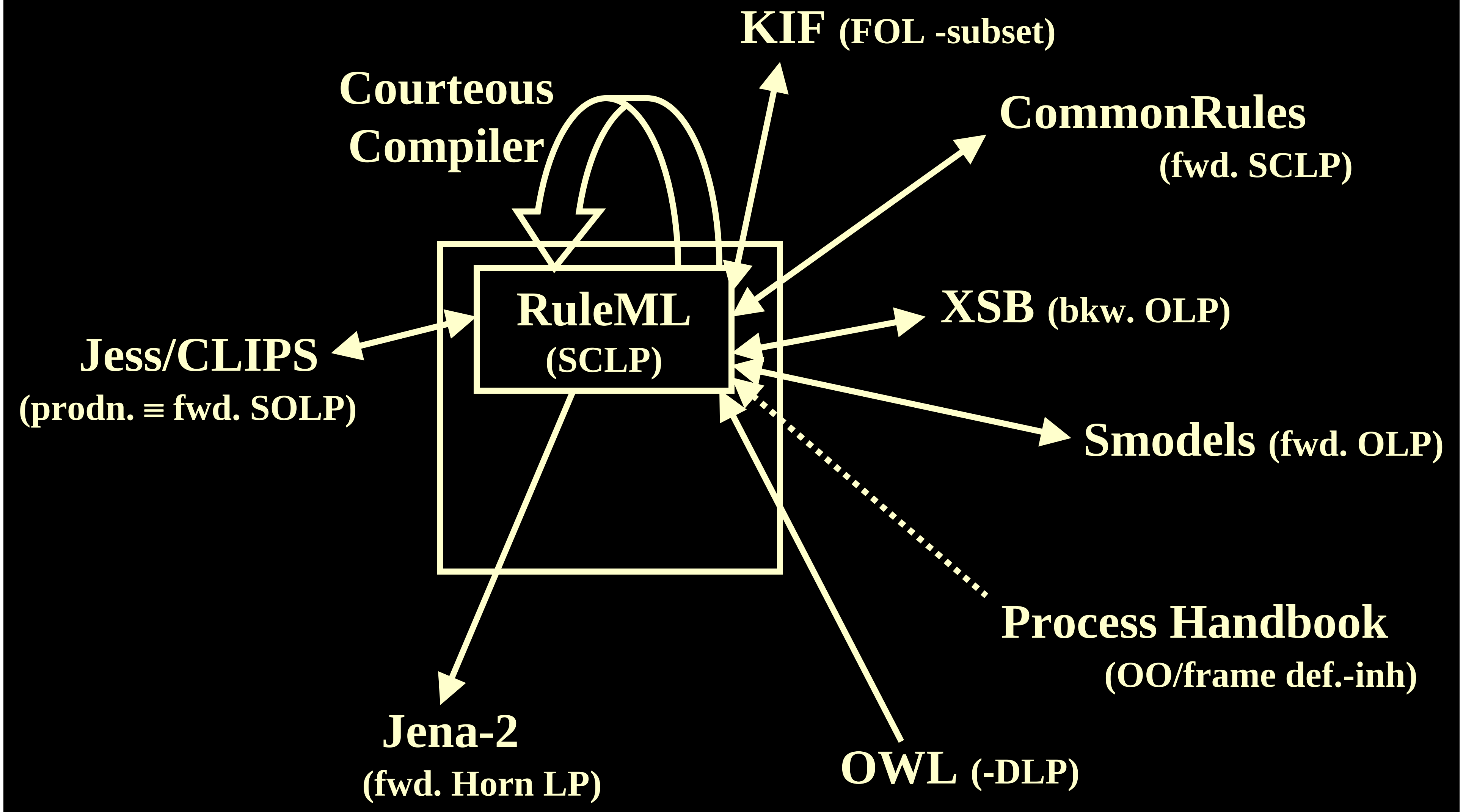
SweetRules Overview

- Concept and Architecture: Open Source Tools Platform for SW Rules and RuleML. http://sweetrules.projects.semwebcentral.org (2004-)
    - Multi-institutional collaboration led by MIT Sloan, with 12+ other co.'s / univ.'s
- Capabilities:
    - Translation and interoperability between heterogeneous rule systems (forward- and backward-chaining) and their rule languages/representations of the most commercially important flavors (relational database / Prolog and production rules / event-condition-action)
    - Inferencing including via translation between rule systems
    - Authoring, Analysis, and testing of rulebases
    - Open, lightweight, extensible, pluggable architecture overall
    - Merge knowledge bases
        - Combine rules with ontologies, incl. OWL, OO default inheritance
    - Focus on kinds of rule systems that are commercially important
        - E.g., Jess production rules, XSB Prolog, IBM Common Rules, HP Jena, …
    - Highly scaleable performance by piggybacking on mature commercial implementations (e.g., Jess, XSB)
    - Automatically composes translators, inference engines

SweetRules V2.0 Fundamental KR

- Fundamental KR: Situated Courteous Logic Programs (SCLP)   KR = Knowledge Representation
  - Horn
  - + Negation-As-Failure (NAF) = Ordinary LP
  - + Courteous prioritized conflict handling
    - overrides relation on rule labels, classical negation, mutex integrity constraints
  - + Situated sensing & effecting
    - Invoke external procedural attachments
    - Sensing = tests/queries; e.g., built-ins
    - Effecting = side-effectful actions, triggered by conclusions

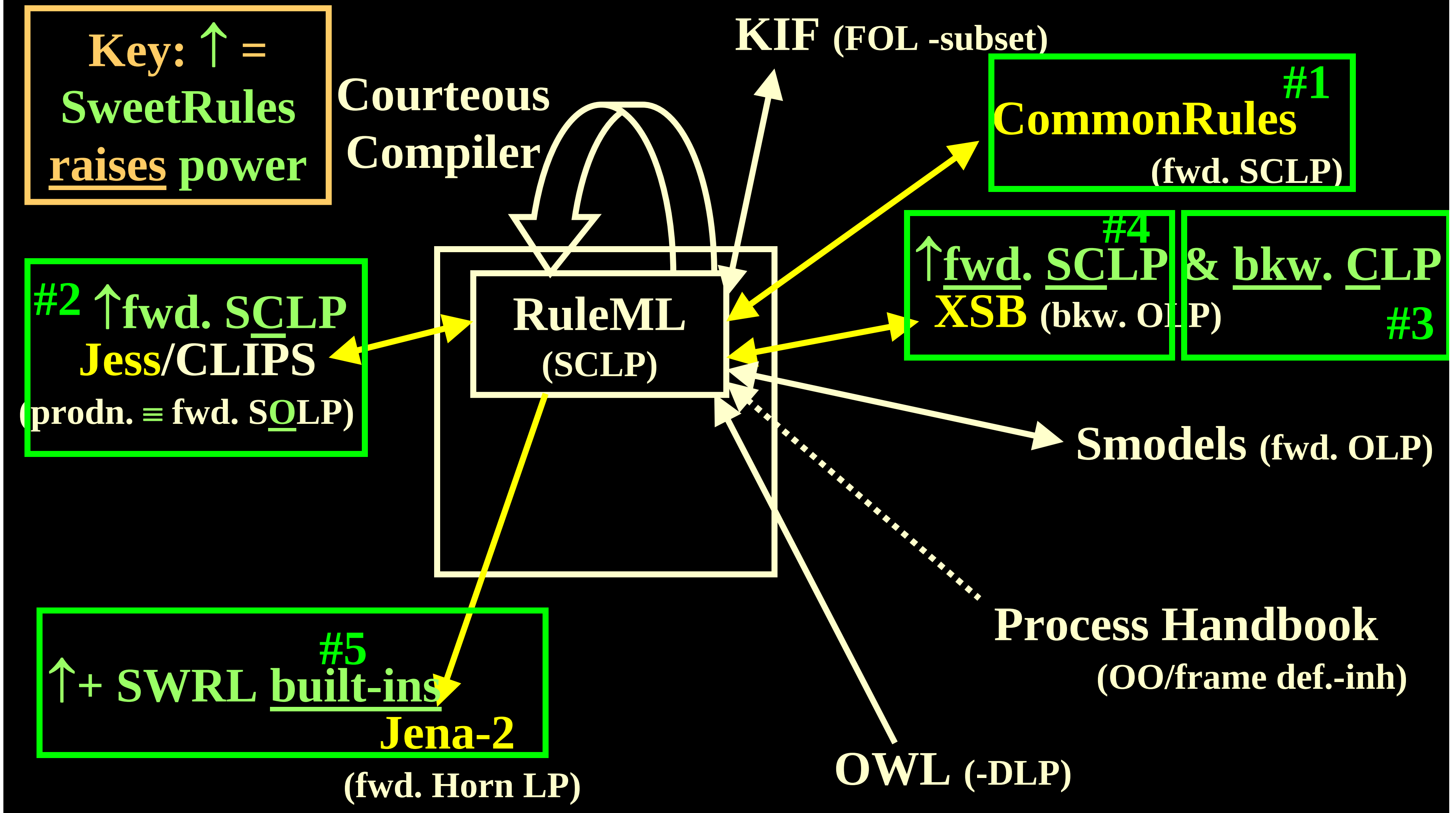
SweetRules V2.0   Translators Graph



SweetRules Inferencing Capabilities Today: Overview

- Inferencing engines in RuleML/SWRL via translation:
  - Indirect inferencing:
    1. translate to another rule system, e.g., {XSB, Jess, CommonRules, or Jena}
    2. run inferencing in that system's engine
    3. translate back
  - Can use composite translators

OPTIONAL: SweetRules V2.0 New Inferencing Engines


Key: ↑ = SweetRules raises power
Courteous Compiler
KIF (FOL -subset)
#1 CommonRules (fwd. SCLP)
#4 ↑fwd. SCLP XSB (bkw. OLP)
& bkw. CLP #3
#2 ↑fwd. SCLP Jess/CLIPS (prodn. ≡ fwd. SOLP)
RuleML (SCLP)
Smodels (fwd. OLP)
Process Handbook (OO/frame def.-inh)
#5 ↑+ SWRL built-ins Jena-2
(fwd. Horn LP)
OWL (-DLP)

Novel NAF Capability in Production Rules I

- Newly Supports Correct Negation-As-Failure in Production Rules
  - Problem: Jess does not correctly implement Negation-As-Failure
    - Conjecture: this problem is shared by all current production rule systems (OPS5-heritage family, based on Rete)
      - *Currently investigating this conjecture.*
  - Solution: We have developed two new techniques with associated KR proof/model theory
    - Stratified case of NAF: declare stratification-based salience in the production rules, when translating from RuleML
      - *Is implemented in SweetRules V2.0 (SweetJess component). Works correctly in all initial phase tests. More testing is in progress.*

Novel NAF Capability in Production Rules II

- General non-stratified case of NAF: new bottom-up algorithm for well founded semantics of OLP
  - *Currently detailed algorithm has been designed and is being implemented.*

- Observation on Additional Value-add: This eliminates the need for agenda meta-rules hacking to get NAF right in production rules, which is frequent in existing production rule applications (and is part of training/methodology)
  - *Interesting Question: How big a percentage of overall agenda meta-rules in typical applications are thus eliminated? Most?*

More Novel Capabilities

- Newly Uses Courteous Compiler to support Courteous feature (prioritized conflict handling) even in systems that don't directly support it, as long as they support negation-as-failure
  - E.g., in XSB Prolog, Jess, Smodels
  - Uses Native Open-Source Courteous Compiler (CC) or CC from IBM CommonRules
- New Include-a-KB mechanism, similar to owl:imports Has Include-a-KB mechanism, similar to owl:imports (prelim. RuleML V0.9)
  - Include a remote KB that is <u>translatable</u> to RuleML
- Uses New Action Launcher component to support Situated effecting feature (actions triggered by conclusions) even in systems that don't directly support it. Facts input, actions output.
  - E.g., in SweetXSB forward inferencing

Novel KB Merging of Rules + Ontologies

- Combine:
  - Multiple SCLP RuleML (/ SWRL) rulebases
    - Or any knowledge base that is translatable into RuleML
  - Heterogeneous kinds of rules
    - E.g., originally XSB rules + Jess facts
    - These get translated and union'd into a single RuleML rulebase (possibly virtual)
  - OWL ontologies
    - Translate Description Logic Programs (DLP) subset of OWL into RuleML
    - Hybrid reasoning via DLP-fusion, i.e., LP inferencing after translate
  - OO/Frame ontologies with default inheritance
    - E.g., Process Handbook ontologies
    - … which get translated to (S)CLP rules

Objectives for Integrating Distributed SW Rules and Ontologies, Motivating SweetRules and its underlying theory+standards

BEFORE

AFTER

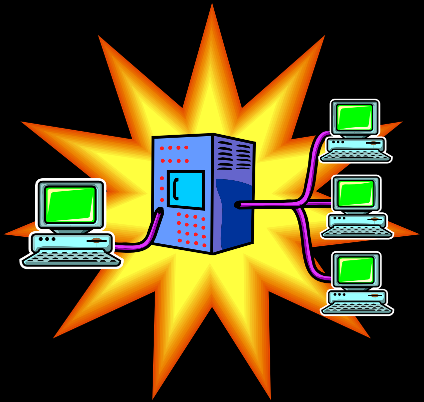
Contradictory conflict is globally contagious, invalidates all results.

$\Rightarrow$

Contradictory conflict is contained locally, indeed tamed to aid modularity.

Knowledge integration tackling the 5 D's (diversity, distributedness, disagreement, dynamism, & delay) is labor-intensive, slow, costly.

$\Rightarrow$



Knowledge integration is highly automated, faster, cheaper.

Outline

- Introduction
  - Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

“Ontology” More General than OWL

- “ontology” in general sense = definitional knowledge [sense from AI and philosophy]
  - Could be in any KR, e.g., FOL, LP, or probabilistic

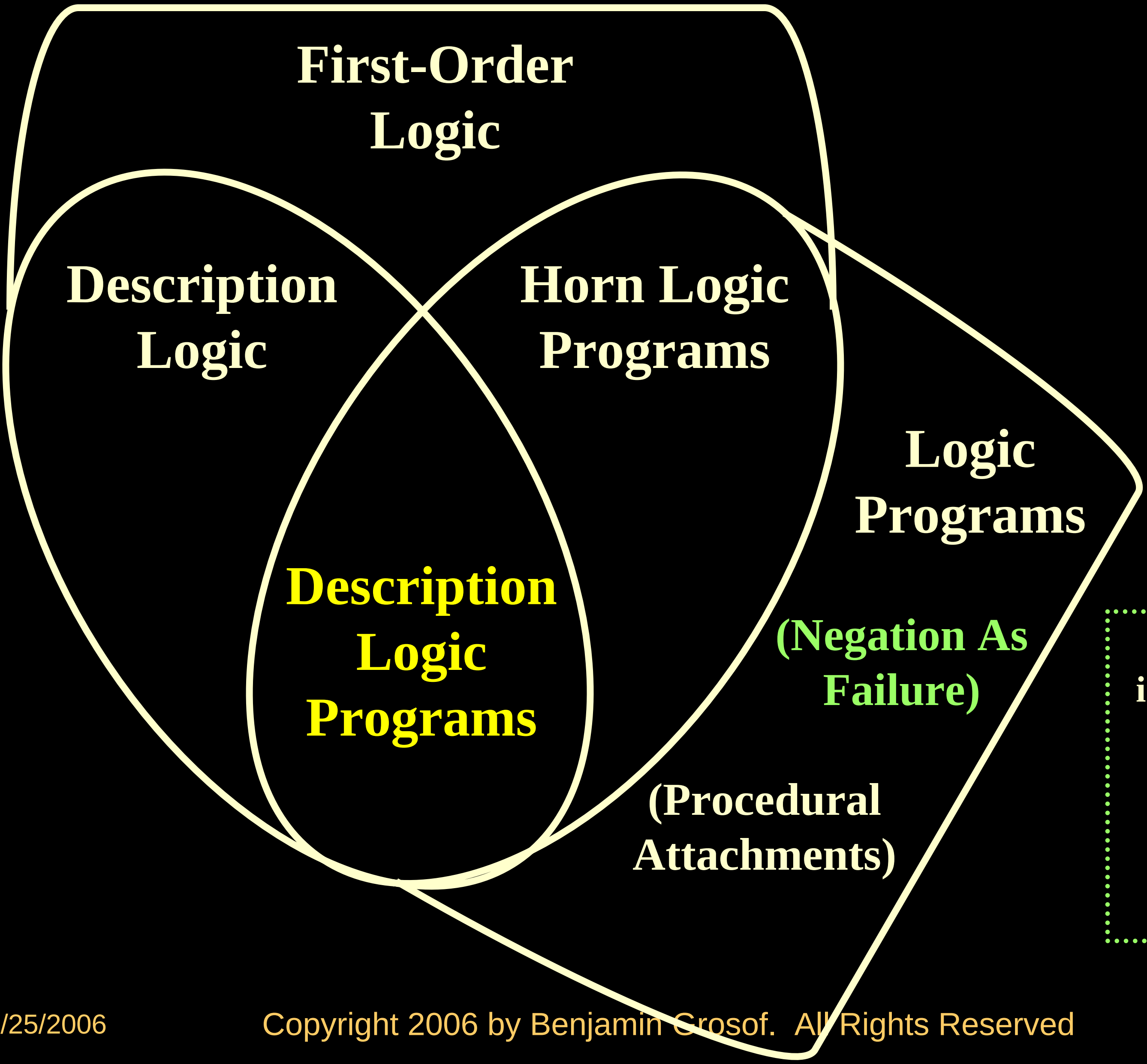
- Important kinds of ontologies:
  - Taxonomies: vocabulary and basic class hierarchy
  - Description Logic
  - Object oriented with default inheritance, e.g., C++/Java/C# class-hierarchy frameworks with overriding or cancellation of inheritance
  - Database schemas
  - XML schemas
  - UML aspects
  - Axiomatizations in FOL of time, space, processes

Aspiration: Unifying FOL and Nonmon LP

- A challenge, a holy grail:
  - Wouldn't it be nice to have a single Knowledge Representation (KR) that unifies <u>all</u> of FOL and nonmon LP?
  - … or at least <u>more</u> of FOL and nonmon LP?

- Physics analogy: "A unified field theory for Semantic Web KR"

Venn Diagram: Expressive Overlaps among KR's



NB: Nonmon LP, including Courteous, relies on NAF as fundamental underlying KR expressive mechanism

Motivations I: Some Potential Uses for Unifying FOL and Nonmon LP KR's for Rules+Ontologies

- Tightly integrate full OWL ontologies (OWL-DL and OWL-Full) with nonmon LP rules. Increase expressiveness of DLP to all of OWL.
  - Semantics; algorithms; ensure consistency
- Cope robustly with conflict between ontologies, e.g., merging OWL ontologies from many sources
- Permit FOL for ontologies beyond DL/OWL
  - E.g., process models cf. NIST's PSL standard and Semantic Web Services Initiative's SWSL emerging standards proposal (http://www.swsi.org)
  - E.g., ECOIN work on equational ontologies and context integration (http://context2.mit.edu/coin)
- Integrate nonmon frame/OO ontologies with mon DL/FOL ontologies

Motivations II: Some Potential Uses for
Unifying FOL and Nonmon LP KR's for Rules+Ontologies

- Integrate SWSL's 2 "wings":
  - LP rules language & service-concept ontologies for contracts, policies, ads, mappings, etc. (SCAMP tasks)
  - FOL language & service-concept ontologies for process model, synthesizing composition, verification, etc. (e.g., cf. PSL)
    - Actually also desire default reasoning to minimize ramifications in reasoning about actions (e.g., cf. Golog)

- Unify the KR foundation of the Semantic Web
  - Represent all the current* major pieces:
    - Rules, ontologies, databases, RDF, queries
    - Semantic Web Services service descriptions
  - Overcome what has been a major hang-up for Joint Committee and Semantic Web Services Initiative efforts on SW standards design.

(*NB: SW in future should also include probabilistic/statistical KR.)

Hypermonotonic Reasoning: Overview

- Definition: A KR S is "hyper"monotonic relative to FOL when S is nonmonotonic and S is sound but incomplete relative to FOL.
  - Premises (conclusions) of S are *viewable as premises (conclusions) of FOL.
  - Generalization: *Under a mapping T from premises/conclusions of S to premises/conclusions of FOL.
- The hypermon KR's entailed conclusions can be viewed as always unobjectionable, i.e., sanctioned, by FOL which provides a background "reference" semantics for the premises in the hypermon KR.

Hypermon: Discussion of Definition

- The spirit of <u>conflict handling</u> is a good match to the hypermon concept.
  - When P is <u>inconsistent</u> according to FOL, then it's arguably often quite <u>desirable</u> that S is incomplete wrt FOL, since FOL produces a global meltdown in which all sentences are entailed.
  - Even if P is <u>consistent</u> according to FOL, then it's <u>"not so bad"</u> that S is incomplete. In practical inferencing over FOL, since that is computationally and/or algorithmically complex, incompleteness is often acceptable. I.e., many practical FOL tools are (in general) incomplete.
    - The hypermon KR can be viewed as a semantically characterized class of incomplete FOL reasoning tools.

- <u>Analogy: jumping through hyperspace</u> (similar to "hyper"text)
  - Overcomes the apparent barrier/limitation of how inconsistency behaves (global fragilility/propagation) in classical logic. "Tunnels through a wormhole" to a consistent, typically contentful, set of conclusions (with localized propagation scope for unresolved conflicts).

Nonmon LP as Hypermon

Caveat: The following results are in preliminary and summary form.

- Obs.: OLP is unsound wrt FOL, if NAF is mapped to classical negation. I.e., Closed World is required as an extra assumption, essentially. Thus OLP is not (directly) hypermon.

- Theorem: NAF-free Courteous LP ("CLP2") is hypermon.
  - (Some other nonmon KR's are too.)

- Theorem: NAF-ful Courteous LP, and thus Ordinary LP, is hypermon under a simple mapping T1:
  - Replace every NAF'd atom ~p(t) by fp(t), where fp is a new predicate.
  - Add the two rules:
    a. fp(t) $\leftarrow$.
    b. $\neg$fp(t) $\leftarrow$ p(t).

Nonmon LP as Hypermon, cont.'d

- Theorem: CLP is always consistent from the viewpoint of FOL. (I.e., it has a consistent set of conclusions.)

- Can thus view conflictful merging/updating in CLP2 as sound, consistent, and incomplete from FOL viewpoint.

- The fundamental KR relationships can be used in more ways too:
  - Import FOL axioms (e.g., ontologies) to become (nonmon) LP rules, mutex's
    - As LP premises
      - E.g., as initial rules or as dynamically sensed facts
  - Export (nonmon) LP conclusions as facts to become FOL axioms
    - An early usage: provide KR semantic analysis of Rei as CLP rules conservatively extending (non-Horn-expressible) DL.

Nonmon LP as Hypermon wrt FOL, cont.'d yet more

- Provides path to formally define and investigate:
  - Merging of LP KB's with FOL KB's, in terms of conclusions or premises, when conflict is absent or present.

- *Further Results in Development, e.g.:*
  - Special cases when (nonmon) LP is consistent, or its updates are monotonic, wrt a given FOL or LP sub-theory/background-theory.
    - E.g., ∃x.q(x) in FOL is consistent with CLP in which all rules with q in head mention q positively. E.g., Rei rules consistent with the ontologies it uses.
  - Identify, tweak, extend, design hypermon KR's

Hypermon Application in SWSF: Translate FOL Ontologies to Courteous LP

- SWSF defines Core Service Ontologies in FOL (not DL).
  - E.g., PSL which is fairly general-form FOL
- Challenge: want an LP version of these ontologies
- Experimental approach based on hypermonotonic reasoning [Grosof] was used to create this version in SWSF 1.0, including for all of PSL Core & Outer Core. Maps general FOL to Courteous LP.
  - Automatable, algorithmic technique, initially performed manually in SWSF 1.0 [Grosof & Martin]:
    1. Skolemize. This produces clausal-form FOL (skolem normal form).
    2. Map each clause into the "omnidirectional set" of LP rules

       (L1 or L2 or … or Lk)

       $\Rightarrow\Rightarrow$

       L1 :- neg L2 and … and neg Lk.

       L2 :- neg L1 and neg L3 and … and neg Lk.

       …

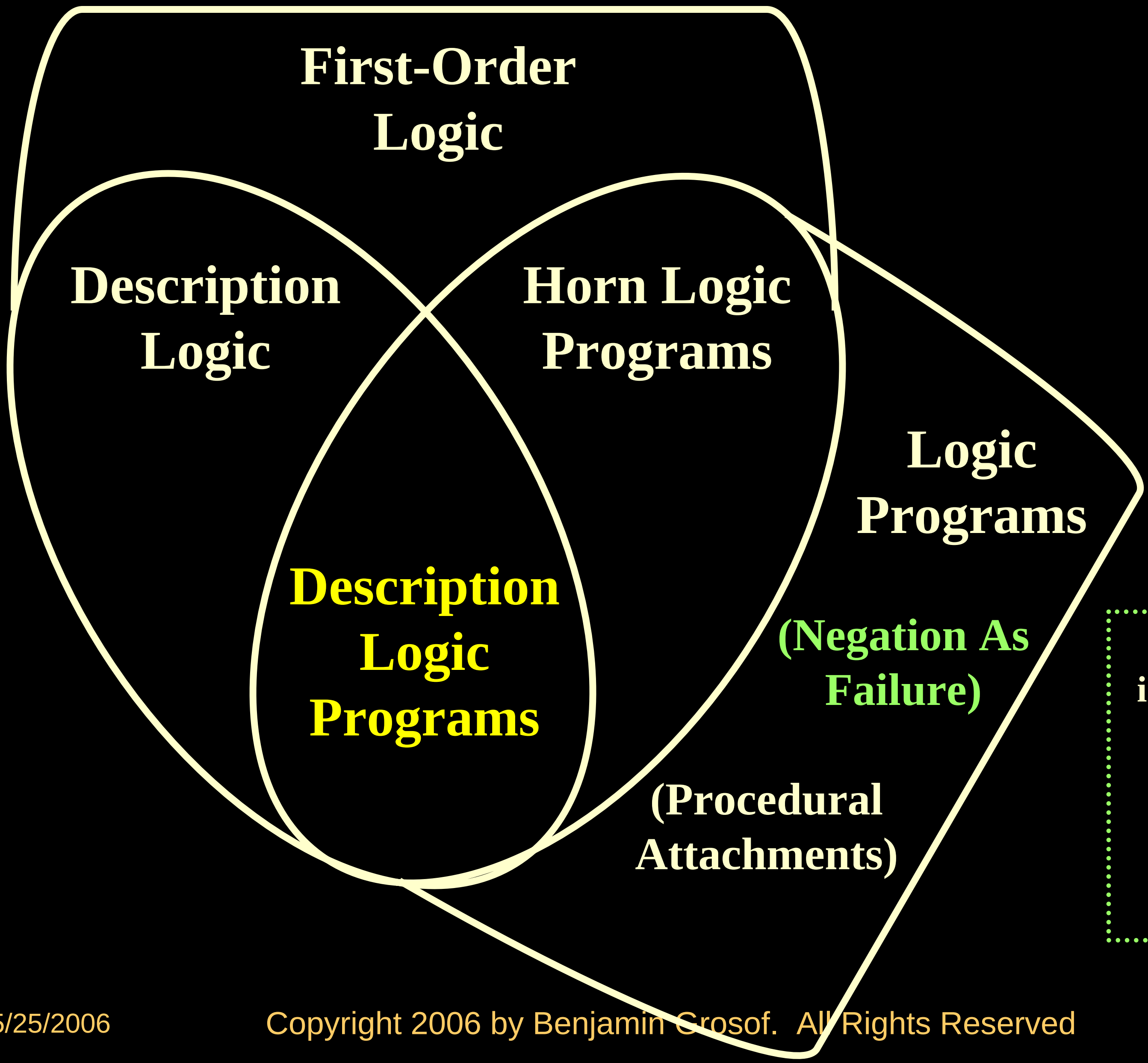
       Lk :- neg L1 and neg L2 and … and neg Lk-1.

Hypermon Application #2 in Progress: *Provide Semantics for Rei*

- Rei 2.0 policy language [Kagal *et* al. 2004]:
  - prioritized default policy rules for authorization
  - on top of OWL ontologies
  - *Aimed at Semantic Web and SWS…*
  - *…BUT: Lacks KR semantics!*
- Approach to providing semantics, *in current work [Grosof & Kagal]:*

1. Represent policy rules via Courteous LP
   - Including "meta-policies" and "meta-meta-policies": represent them as reasoning about prioritization
2. Hypermonotonic reasoning to integrate the KR semantics of: the policy rules + the ontologies
   - Exploit Rei's restriction that:
     - OWL-defined predicates appear only in *bodies* of policy rules
   - Analyze composite semantics in terms of conservative extension

Venn Diagram: Expressive Overlaps among KR's



NB: Nonmon LP, including Courteous, relies on NAF as fundamental underlying KR expressive mechanism

Motivations I: Some Potential Uses for Unifying FOL and Nonmon LP KR's for Rules+Ontologies

- Tightly integrate full OWL ontologies (OWL-DL and OWL-Full) with nonmon LP rules. Increase expressiveness of DLP to all of OWL.
  - Semantics; algorithms; ensure consistency
- Cope robustly with conflict between ontologies, e.g., merging OWL ontologies from many sources
- Permit FOL for ontologies beyond DL/OWL
  - E.g., process models cf. NIST's PSL standard and Semantic Web Services Initiative's SWSL emerging standards proposal (http://www.swsi.org)
  - E.g., ECOIN work on equational ontologies and context integration (http://context2.mit.edu/coin)
- Integrate nonmon frame/OO ontologies with mon DL/FOL ontologies

Motivations II: Some Potential Uses for Unifying FOL and Nonmon LP KR's for Rules+Ontologies

- Integrate SWSL's 2 "wings":
  - LP rules language & service-concept ontologies for contracts, policies, ads, mappings, etc. (SCAMP tasks)
  - FOL language & service-concept ontologies for process model, synthesizing composition, verification, etc. (e.g., cf. PSL)
    - Actually also desire default reasoning to minimize ramifications in reasoning about actions (e.g., cf. Golog)

- Unify the KR foundation of the Semantic Web
  - Represent all the current* major pieces:
    - Rules, ontologies, databases, RDF, queries
    - Semantic Web Services service descriptions
  - Overcome what has been a major hang-up for Joint Committee and Semantic Web Services Initiative efforts on SW standards design.

(*NB: SW in future should also include probabilistic/statistical KR.)

Outline

- Introduction
  - Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

Impacts: Translation of LP/RuleML ↔ Production Rules

- Part of what's now called Production LP
- Uses Situated LP for actions and tests
  - Situated LP an extension feature for LP, adopted by RuleML

- A focus in SW rules R&D community; many implementing translation of LP → Jess
- Underpins agenda/optimism/energy in W3C RIF that production rule vendors join the SW

Some Technical Directions for Research

- Incremental Reasoning: Events, Updates
- LP KR other extensions:
  - Existentials via skolemization
  - Combine Hilog higher-order features reducible to first-order; OWL-Full, RDF-Full
  - Equality: user-defined, nonmonotonic
  - Reification
- Hypermonotonicity: analysis of LP, merging; new KR's incl. disjunctive
- Probabilistic, decision-theoretic, game-theoretic; Inductive, learning, data mining
- Constraints: satisfaction, optimization

- Trust policies for firewalls, confidentiality, security, privacy, access control
- E-Contracting end-to-end reuse, power: incl. business process monitoring
- Policy Ontology, Services Ontologies, Relationship to C++/Java/C# Inheritance
- Web Services "Policy Management", "Contracts"
- Add semantics to existing standards: XBRL, XACML, ebXML, RosettaNet, EDI
- Biomedical: patient records privacy and workflow, drug discovery, treatment safety tracking
- Marketing, intelligence, supply chain, financial reporting, travel
- Business Value Analysis, Strategy, Roadmapping

OPTIONAL SLIDES BEGIN

EECOMS Example of SCM Policy Rules: Ordering Lead Time

- Vendor's rules that prescribe how buyer must place or modify an order:
- A) 14 days ahead if the buyer is a qualified customer.
- B) 30 days ahead if the ordered item is a minor part.
- C) 2 days ahead if the ordered item's item-type is backlogged at the vendor, the order is a modification to reduce the quantity of the item, and the buyer is a qualified customer.

- Suppose more than one of the above applies to the current order? **Conflict!**

- Helpful Approach: **precedence** between the rules. Often only *partial* order of precedence is justified. E.g., C > A.

Courteous LP's: Ordering Lead Time Example

```
{leadTimeRule1} orderModificationNotice(?Order,14days)
                       ←  preferredCustomerOf(?Buyer,?Seller) ∧
                          purchaseOrder(?Order,?Buyer,?Seller) .
{leadTimeRule2} orderModificationNotice(?Order,30days)
                    ← minorPart(?Buyer,?Seller,?Order) ∧
                      purchaseOrder(?Order,?Buyer,?Seller) .
{leadTimeRule3} orderModificationNotice(?Order,2days)
                  ←   preferredCustomerOf(?Buyer,?Seller) ∧
                      orderModificationType(?Order,reduce) ∧
                      orderItemIsInBacklog(?Order) ∧
                      purchaseOrder(?Order,?Buyer,?Seller) .
overrides(leadTimeRule3 , leadTimeRule1) .
⊥ ← orderModificationNotice(?Order,?X) ∧
           orderModificationNotice(?Order,?Y) |  (?X ≠?Y) .
```

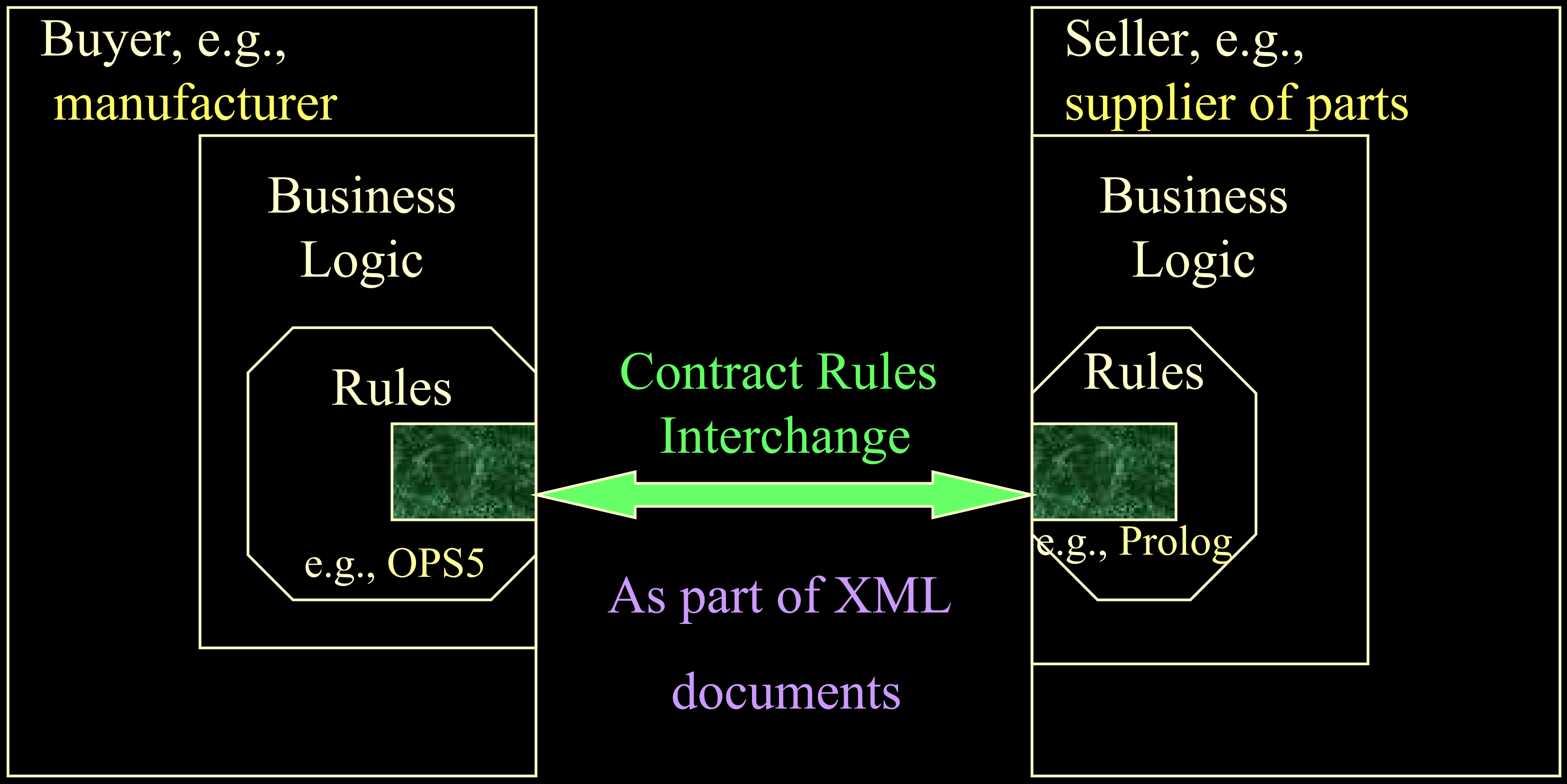
End-to-End E-Contracting Tasks

- Discovery, advertising, matchmaking
  - Search, sourcing, qualification/credit checking
- Negotiation, bargaining, auctions, selection, forming agreements, committing
  - Hypothetical reasoning, what-if'ing, valuation
- Performance/execution of agreement
  - Delivery, payment, shipping, receiving, notification
- Problem Resolution, Monitoring
  - Exception handling

SweetDeal Approach: Rule-based Contracts for E-commerce

- Rules as way to specify (part of) business processes, policies, products: as (part of) contract terms.
- Complete or partial contract.
  - As default rules. Update, e.g., in negotiation.
- Rules provide high level of conceptual abstraction.
  - easier for non-programmers to understand, specify, dynamically modify & merge. E.g.,
  - by multiple authors, cross-enterprise, cross-application.
- Executable. Integrate with other rule-based business processes.

Contract Rules during Negotiation

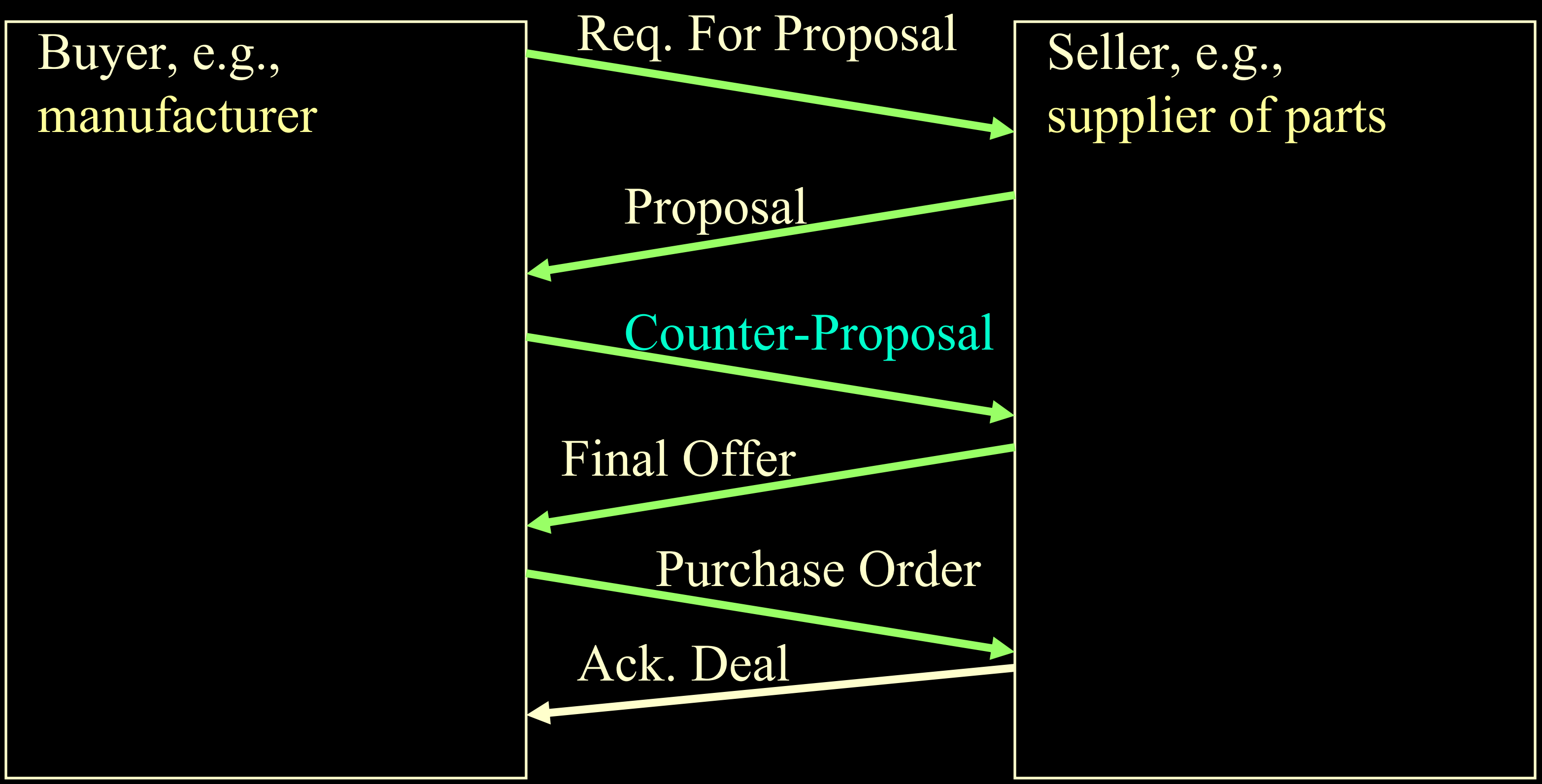


Contracting parties NEGOTIATE via shared rules.

Examples of Contract Provisions Well-Represented by Rules in Automated Deal Making

- Product descriptions
  - Product catalogs: properties, conditional on other properties.
- Pricing dependent upon: delivery-date, quantity, group memberships, umbrella contract provisions
- Terms & conditions: refund/cancellation timelines/deposits, lateness/quality penalties, ordering lead time, shipping, creditworthiness, biz-partner qualification, service provisions
- Trust
  - Creditworthiness, authorization, required signatures
- *Buyer Requirements (RFQ, RFP) wrt the above*
- *Seller Capabilities (Sourcing, Qualification) wrt the above*

Exchange of Rules Content during Negotiation: example



Example: E-Contract Proposal from supplierCo to manufCo

- …
  {usualPrice} price(per_unit, ?PO, \$60) $\leftarrow$
- purchaseOrder(?PO, supplierCo, ?AnyBuyer) $\wedge$
- quantity_ordered(?PO, ?Q) $\wedge$ (?Q $\geq$ 5) $\wedge$ (?Q $\leq$ 1000) $\wedge$
- shipping_date(?PO, ?D) $\wedge$ (?D $\geq$ 24Apr00) $\wedge$ (?D $\leq$ 12May00).
- {volumeDiscount} price(per_unit, ?PO, \$51) $\leftarrow$
- purchaseOrder(?PO, supplierCo, ?AnyBuyer) $\wedge$
- quantity_ordered(?PO, ?Q) $\wedge$ (?Q $\geq$ 100) $\wedge$ (?Q $\leq$ 1000) $\wedge$
- shipping_date(?PO, ?D) $\wedge$ (?D $\geq$ 28Apr00) $\wedge$ (?D $\leq$ 12May00) .
  overrides(volumeDiscount , usualPrice) .
- $\perp$ $\leftarrow$ price(per_unit, ?PO, ?X) $\wedge$ price(per_unit, ?PO, ?Y) GIVEN (?X $\neq$?Y).
- ...

Negotiation Ex. Doc. Rules: Counter-Proposal *from manufCo to supplierCo*

- …
  {usualPrice} price(per_unit, ?PO, \$60) ← …
- {volumeDiscount} price(per_unit, ?PO, \$51) ←
- purchaseOrder(?PO, supplierCo, ?AnyBuyer) ∧
- quantity_ordered(?PO, ?Q) ∧ (?Q ≥ 5) ∧ (?Q ≤ 1000) ∧
- shipping_date(?PO, ?D) ∧ (?D ≥ 28Apr00) ∧ (?D ≤ 12May00) .
  overrides(volumeDiscount , usualPrice) .
- ⊥ ← price(per_unit, ?PO, ?X) ∧ price(per_unit, ?PO, ?Y) GIVEN (?X ≠ ?Y).
- {aSpecialDeal} price(per_unit, ?PO, \$48) ←
- purchaseOrder(?PO, supplierCo, manufCo) ∧
- quantity_ordered(?PO, ?Q) ∧ (?Q ≥ 400) ∧ (?Q ≤ 1000) ∧
- shipping_date(?PO, ?D) ∧ (?D ≥ 02May00) ∧ (?D ≤ 12May00) .
- overrides(aSpecialDeal, volumeDiscount) .
- overrides(aSpecialDeal , usualPrice) .
- …

Simply *added* rules!

Negotiation Example -- XML Encoding of Rules in RuleML

- <rulebase>
-   <imp>
-     <rlab>usualPrice</_rlab>
-     <head>
-       <atom>
-         <opr><rel>price</rel></_opr>
-           <ind>per_unit</ind>
-           <var>PO</var>
-           <ind>$60</ind>
-       </atom>
-     </head>
-     <body>   … *(see next page)*   </_body>
-   </imp>
- …
- </rulebase>

What Can Be Done with the Rules in contracting, & negotiation, based on our SweetDeal approach to rule representation

- Communicate: with deep shared semantics
  - via RuleML, inter-operable with same sanctioned inferences
  - ⇔ heterogeneous rule/DB systems / rule-based applications ("agents")
- Execute contract provisions:
  - infer; ebiz actions; authorize; ...
- Modify easily: contingent provisions
  - default rules; modularity; exceptions, overriding
- Reason about the contract/proposal
  - hypotheticals, test, evaluate; tractably
  - *(also need "solo" decision making/support by each agent)*

Example of Entailment: Sunday Stroll

- In Bayesian Probability KR:
  - Let P be the premises:
    - prob(rainySunday) = 0.4.
    - prob(funSunday | rainySunday) = 0.3.
    - prob(funSunday | ¬rainySunday) = 0.9.
  -
  - In this KR, P entails (among others) the conclusion:
    - prob(funSunday) = 0.66.

Outline

- Introduction
  - Commercial Rule Systems, Standards, Semantic Web, Services, Business Applications, Business Value
    - Semantic Interoperability Challenges
  - Production LP Approach to KR
- Production LP KR Extension/Feature
- + Default Negation (Negation As Failure) Feature
- + Courteous Feature; + Other Features
- Semantic Translation of Production Rules ↔ PLP
- SweetRules implementation, translation ↔ Jess
- + Ontologies: Description LP; OO default inheritance
  - FOL (beyond DLP) via hypermonotonic reasoning
- Conclusions and Future Work

OPTIONAL SLIDES END