

MathServe

—

Brokering of Semantic Reasoning Web Services

Jürgen Zimmer

Universität des Saarlandes, Germany.

This work was partly supported by the European Union
CALCULEMUS IHP Training Network HPRN-CT-2000-00102

Motivation & Background

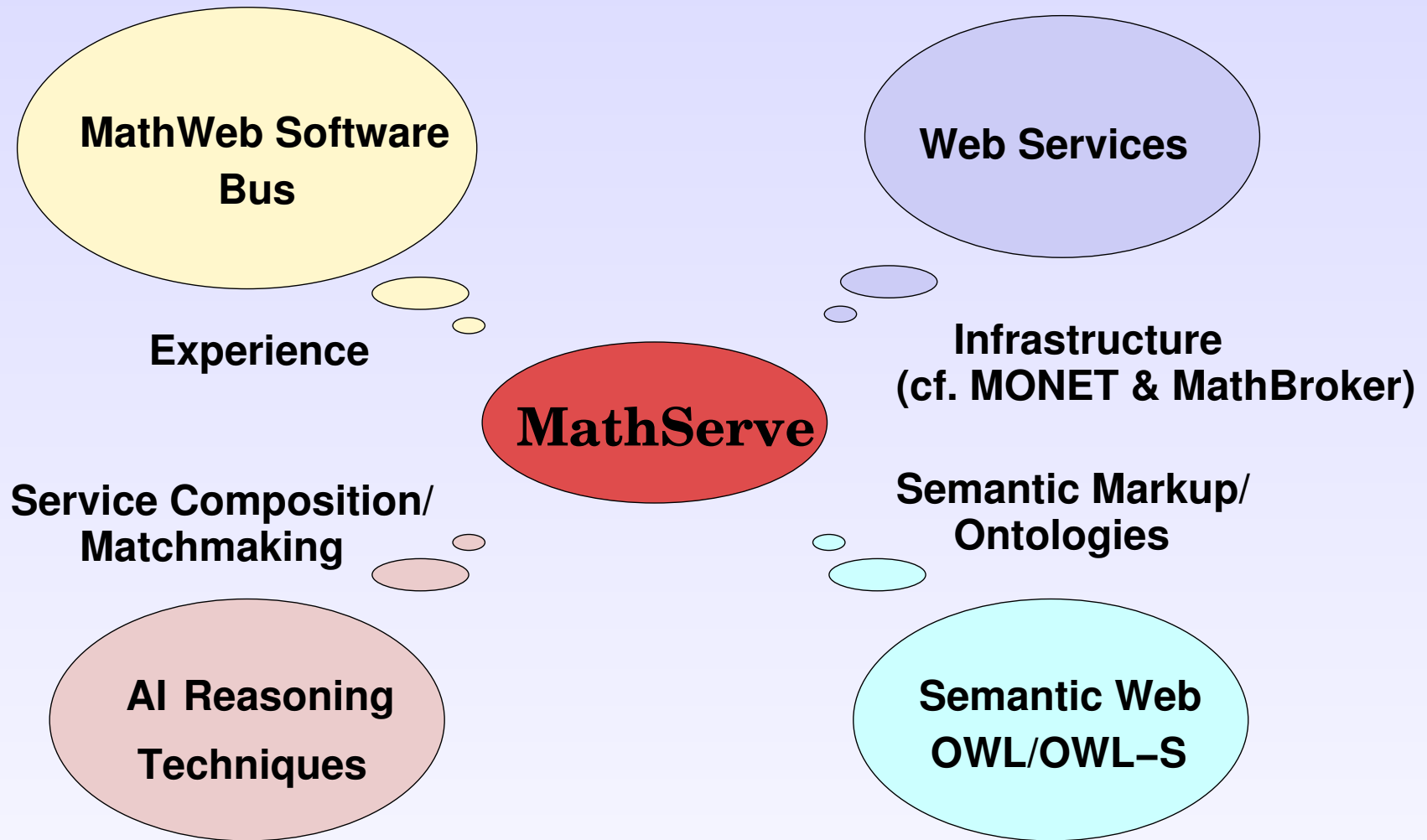
Many **automated reasoning systems** are currently available:

- Automated Theorem Proving Systems.
- Proof/Problem transformation systems.
- Decision Procedures.
- SAT solvers.

Problem: Systems are **difficult to use** for
humans and machines.

Goal: Make systems **interoperable** and available for
automated retrieval and composition.

The MathServe Framework



MathServe in Four Steps

In MathServe we ...

- **Integrate** reasoning systems as Web Services.
- Develop a **domain ontology**.
- Describe services' **semantics in OWL-S**.
- Use the semantic markup for **Matchmaking** and **Automated Web Service Composition**.

Systems Integrated as Web Services

1) Automated theorem proving systems:

- EP, Otter, SPASS, Vampire, Waldmeister.
- For classical first-order predicate logic with equality.

2) Problem transformation systems:

- FLOTTER, tptp2X.
- Specialised clause normal form generator.
- Definition expansion modulo a theory (experimental).

Proof Transformation Systems Integrated

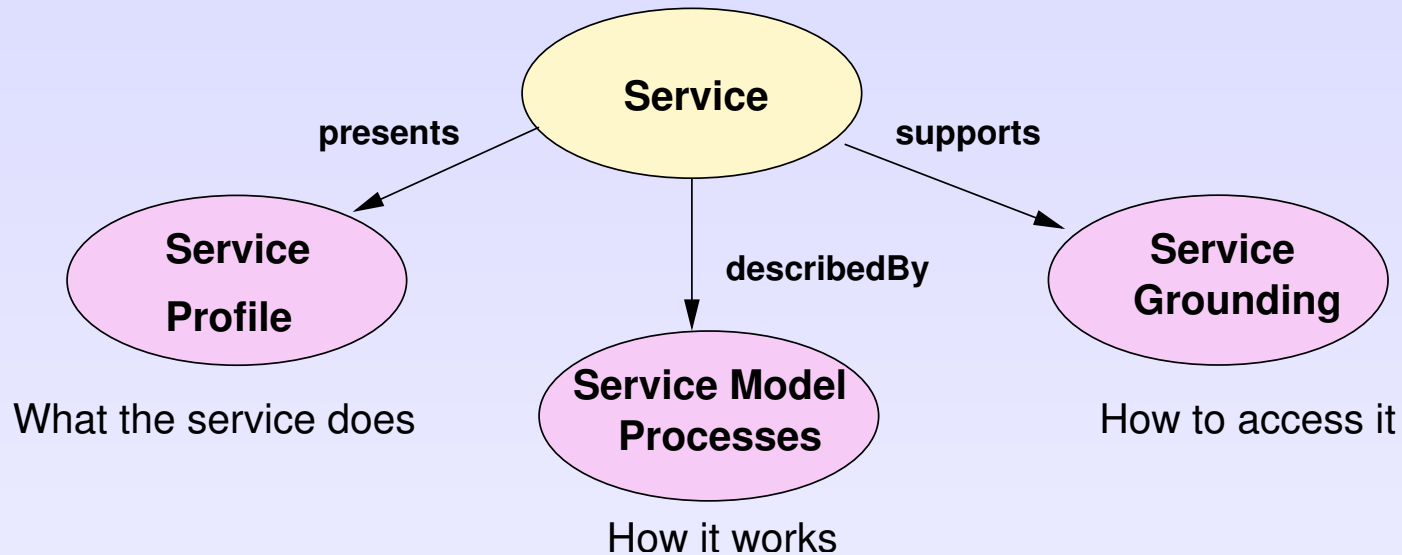
3) **Otterfier Tool** for proof transformation [IJCAR'04-WS]:

- CNF refutation \mapsto CNF refutation (BrFP) calculus
(BrFP = Binary resolution, Factoring, Paramodulation)
- Calls **Otter** to replace “alien” inference steps

4) The **Tramp** system [Meier'00]:

- FOF problem
+
CNF refutation (BrFP) \mapsto ND proof at assertion level.

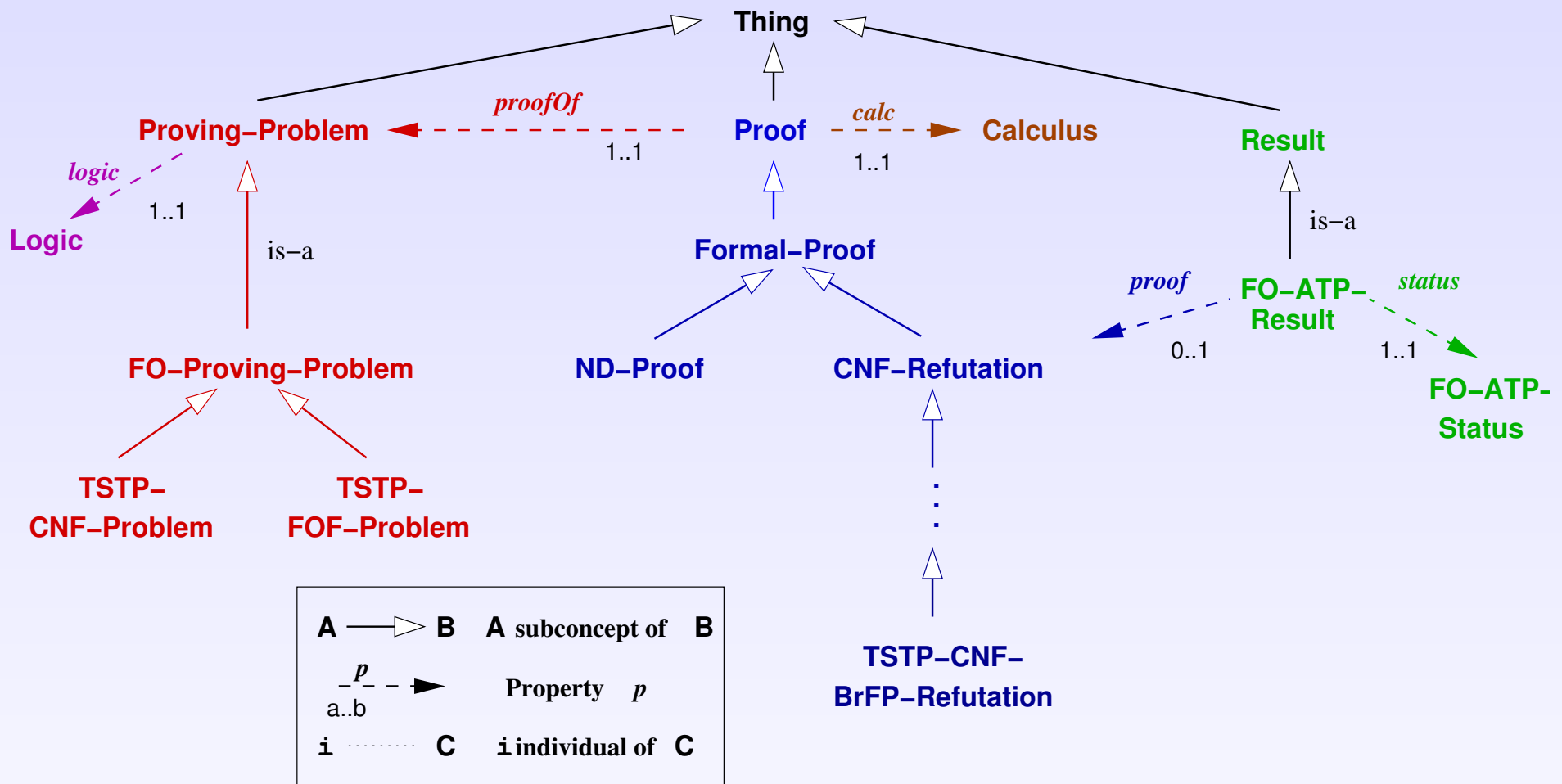
Service Descriptions in OWL-S



- Based on a **domain ontology** in Web Ontology Language (OWL)
- OWL-S processes similar to MSDL* Problems:
 - Inputs and Outputs (IO).
 - Preconditions and Effects (PE).

*MSDL = Mathematical Service Description Language

An OWL Domain Ontology (Fragment)



Example: ATP System Otter

- **Resolution-based ATP** for classical FO-Logic with equality.
- **Result uncertain** (Semi-decidability of FOL).
- **Performance on TPTP Library** (>7000 problems) [Sutcliffe'00]:

Problem Class	Problems solved (%)
FOF_NKC_EPR	55
CNF_NKS_RFO_PEQ_UEQ	73
⋮	⋮

RFO (Real First-Order)

vs. EPR (Essentially Propositional)

CNF (Clause Normal Form)

vs. FOF (First Order Form)

PEQ (Pure Equality)

vs. NEQ (No Equality)

NKC/NKS (Not Known to be (Counter-) Satisfiable)

OWL-S Service Profile for Otter ATP Service

profile **OtterATP**:

inputs: *tstp_problem* – mw# TSTP-CNF-Problem

outputs: *atp_result* – mw# FO-ATP-Result

preconds:

effects: `resultFor(atp_result, tstp_problem)`

categs: `http://www.mathweb.org/owl/categories.owl# FO-ATP`

params: `problemClass(tstp_problem, mw#FOF_NKC_EPR)`

`↪ status(atp_result, stat# Theorem) (0.55) (8009ms)`

`problemClass(tstp_problem, mw#CNF_NKS_RFO_PEQ_UEQ)`

`↪ status(atp_result, stat# Unsatisfiable) (0.73) (2384ms)`

...

`mw = ⟨http://www.mathweb.org/owl/mathserv.owl⟩`

OWL-S Profile of Proof Transformation Service

profile TrampNDforFOF:

inputs: *fof_problem* – mw# TstpFofProblem

atp_result – mw# FOBrFPResult

delta_relation – mw# DeltaRelation

outputs: *nd_proof* – mw# TwegaNDProof

preconds: resultFor(*atp_result*, *cnf_problem*) \wedge

satEquivalentTo(*cnf_problem*, *fof_problem*) \wedge

status(atp_result, mw# Unsatisfiable) \wedge

relatesCNF(*delta_relation*, *cnf_problem*) \wedge

toFOF(*delta_relation*, *fof_problem*)

effects: proofOf(*nd_proof*, *fof_problem*)

catags: [http://www.mathweb.org/owl/categories.owl# Proof-Trafo](http://www.mathweb.org/owl/categories.owl#Proof-Trafo)

params:

mw = \langle <http://www.mathweb.org/owl/mathserv.owl> \rangle

Composition of Semantic Web Services

Problem: Sometimes **one service is not sufficient** to answer a query.

Brokering: Create **composite services**.

Composition constructs in OWL-S:

- **Sequence**
- **Split (+Join)**
- **If-Then-Else**
- **Repeat-Until**
- **Nondeterministic Choice**

Composition Techniques in the Literature

- **AI Planning** [McDermott'02, HendlerEtAl'03, Traverso & Pistore'04,...]
- **Golog** [McIlraith & Son'02]
- **Markov Decision Processes** [Doshi'04]
- **Program Synthesis** [Matskin'02]
- **Linear Logic** [Rao'04 PhD]

Composition in MathServe

Two Step approach:

1. Planning with PRODIGY

- Types of I/O parameters.
- Deterministic effects and static preconditions.

2. Translation of PRODIGY plans into DTGolog programs

- Stochastic effects.
- Sensing actions.
- Optimal choice of actions (services).

Golog = Language based on Situation Calculus.

DTGolog = Golog + Decision Theory (Markov Decision Processes).

Example: Service Composition

Scenario: Brokering of proving/transformation services.

Available Services:

TrampCNF: clause normal form generator.

OtterATP/EpATP/SpassATP: first-order ATP services.

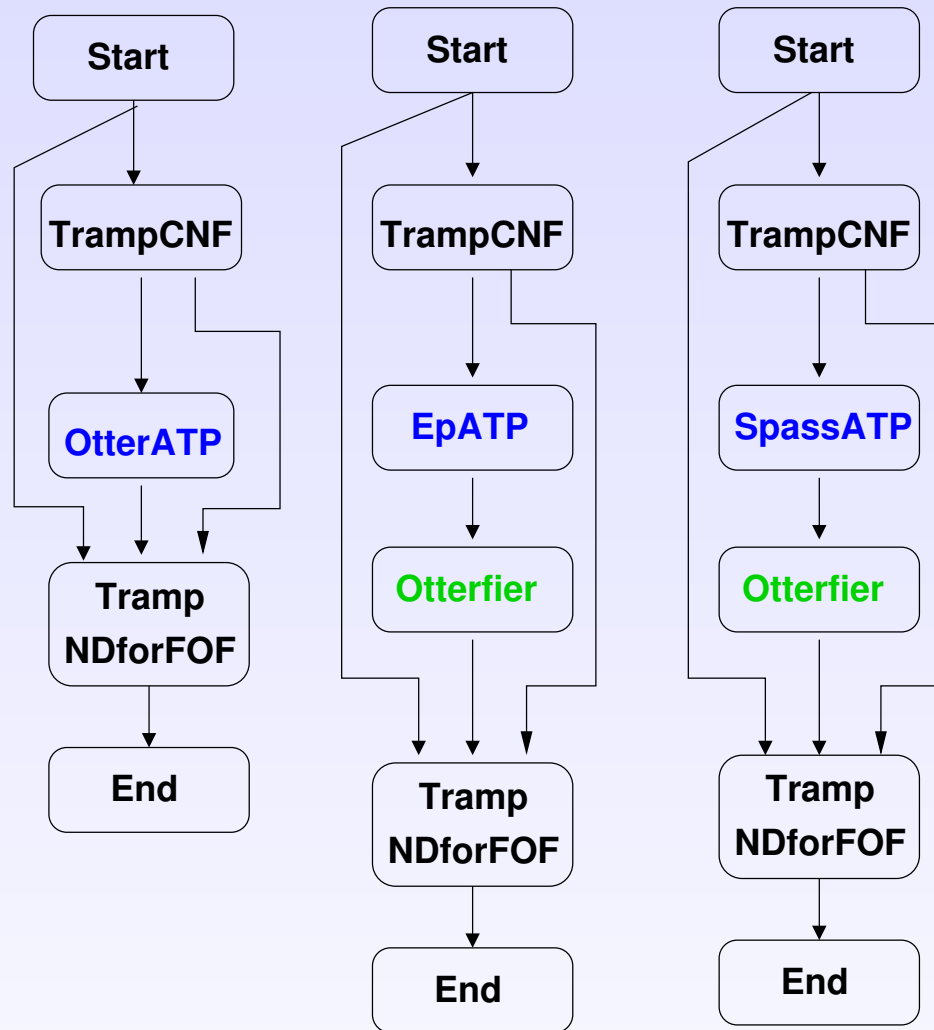
NDforFOF: transforms resolution proofs into ND calculus.

Otterfier: transforms arbitrary resolution proofs
to resolution proofs in restricted Otter calculus.

Query: **Given:** First-order conjecture (FOF) $\Gamma \vdash \psi$.

Return: ND proof object P for $\Gamma \vdash \psi$.

Three PRODIGY Plans for Query



PRODIGY plans in DTGolog

The three PRODIGY plans in one Golog procedure:

```
proc composite(TstpFofProblem, NDProof)
  trampCNF(TstpFofProblem, TstpCnfProblem, DeltaRel) ;
  problemAnalyser(TstpCnfProblem, ProblemClass) ;
  ( (otterATP(TstpCnfProblem, ATPResult) ;
    trampNDforFOF(TstpFofProblem, ATPResult, DeltaRel, NDProof))
  | (epATP(TstpCnfProblem, ATPResult) ;
    otterfier(ATPResult, ATPBrfpResult) ;
    trampNDforFOF(TstpFofProblem, ATPBrfpResult, DeltaRel, NDProof))
  | (spassATP(TstpCnfProblem, ATPResult) ;
    otterfier(ATPResult, ATPBrfpResult) ;
    trampNDforFOF(TstpFofProblem, ATPBrfpResult, DeltaRel, NDProof)))
end
```

Optimal Policy in DTGolog

Offline DTGolog interpreter computes optimal policy:

```
proc optimal(TstpFofProblem, NDProof)  
  trampCNF(TstpFofProblem, TstpCnfProblem, DeltaRel) ;  
  problemAnalyser(TstpCnfProblem, ProblemClass) ;  
  if problemClass(TstpCnfProblem, rfo_peq_cnf_nue) then  
    (otterATP(TstpCnfProblem, ATPResult) ;  
    if status(ATPResult, proof) then  
      trampNDforFOF(TstpFofProblem, ATPResult, DeltaRel, NDProof)  
    else stop endif  
  else if problemClass(TstpCnfProblem, epr_neq_fof) then  
    epATP(TstpCnfProblem, ATPResult) ;  
    otterfier(ATPResult, ATPBrfpResult) ;  
  ...  
end
```

Evaluation

- MATHSERVE took part in the **CADE-20 ATP System Competition (CASC) 2005**
- System stable enough for Competition environment.
- Performance with **optimal policy**:

System	Problems given	Problems solved	Percentage of given	Percentage complete
MathServe 0.62	660	392	59.4%	59.4%
Ep 0.9pre3	660	409	62.0%	62.0%
Vampire 8.0	540	430	79.6%	65.2%

600sec time limit for each problem.

MathServe's Potential

After CASC-20 we...

- ... integrated newer versions of provers, and
- ... added two specialised provers (DCTP, Waldmeister)

Performance with **new provers and new optimal policy**:

System	Problems given	Problems solved	Percentage of given	Percentage complete
MathServe 0.71	660	451	68.3%	68.3%
Ep 0.9pre3	660	409	62.0%	62.0%
Vampire 8.0	540	430	79.6%	65.2%

Conclusion

Semantic reasoning web services...

- ... can be described using **standard Semantic Web technology/languages**.
- ... are suitable for **service matchmaking**.

Reasoning service composition...

- ... has to meet **peculiar requirements**.
- ... can be done with a combination of **AI planning and DTGolog**.

Future Work

- Further **evaluation** in interactive theorem proving context.
- **Stateful Web Services?**
(Trade off: Semantics vs. Protocol Complexity)
- **Decisions depending on available time.**
(Clever Assignment time to different services)

OWL-S Compared to MSDL

Differences between OWL-S and MSDL:

	OWL-S	MSDL
Descriptions based on	OWL-DL*	Proprietary XML Language
Conditions Language	Language of Choice†	OpenMath
Support	Fairly Good	Poor
Tools	Available/Emerging	None

* OWL is W3C Recommendation.

† KIF and SWRL recommended.

Nondeterministic Actions in DTGolog

■ Representation of nondeterministic as actions:

```
senseCondition(OtterATPS(TstpProblem, ATPResult), status(ATPResult, proof)).  
senseCondition(OtterATPF(TstpProblem, ATPResult), -status(ATPResult, proof)).  
...
```

■ Probabilities that nature chooses deterministic actions:

```
prob(OtterATPS(TstpProblem, ATPResult), Pr, S) :-  
    holds(problemClass(TstpProblem, fof_nkc_epr), S),      (Pr is 0.55) ;  
    holds(problemClass(TstpProblem, cnf_nks_rfo_peq_ueq), S), (Pr is 0.73)  
    ...
```

■ Rewards and costs of actions.

```
reward(R, do(A, S)) :-  
    ((A = otterATPS(_, _), R1 is 4) ;  
     (A = otterATPF(_, _), R1 is 0) ;  
     ... )  
    R is R1 - Cost.
```