Houdini algorithm

•••

First idea

General idea

The main idea is to build an algorithm that computes the extension of a given defeasible theory maintaining linear (on the number of literals) space and time complexity.

First, it injects facts into all rules and then literals, heads of previously triggered rules, into strict rules, iterating until it reaches the fixed point i.e. no more rules are updated.

Second, it converts all the remaining strict rules into defeasible rules and repeats this process a second time considering only irrefutable literals and maintaining a temporary set of literals that are obtained from triggered rules but are not irrefutable due to the existence of conflicting rules; they may be added at the end if no contradictions are found.

Data structures

- **Rule** :
  id
  type (strict, defeasible, defeater)
  head : Literal
  tail : list Literal
  consumed: bool

- **Theory** :
  Facts : list Literal
  Rules
  Literals : list Literal
  supRelations : list supRelation

- **supRelation** : (r1 : Rule, r2 : Rule), with r1 < r2

- **+Delta, -Delta, +delta, -delta** : set Literal

- **Literals, Irrefutables** : set Literal

- **TmpPar** : set (Literal, Rule)

- **Deactivate** : dictionary with a literal (as label) associated to a list of rules

Main procedure

```
Initialize +Delta <- Theory.Facts
Initialize -Delta <- 0
InjectLiteralsIntoRules(Theory.Facts, Theory.unconsumedRules)
seekStrictRulesFixpoint()
add {Theory.L \ +Delta} to -Delta
Initialize +delta <- +Delta
Initialize -delta, TmpPar <- 0
seekDefeasibleRulesFixpoint()
add TmpPar to +delta
add {Theory.Literals \ +delta} to -delta
return +Delta, -Delta, +delta, -delta
```

Injection procedure

```
InjectLiteralsIntoRules(Literals, Rules):
    Deactivate[l] <- 0 for l in Literals
    for l in Literals:
        if ~empty(Deactivate[l]) then Deactivate[l] <- 0
        for r in Rules:
            if r.head equals l then remove r
            if r.tail contains l then remove l from r.tail
            if r.tail contains ~l then add r to Deactivate[~l]
    deactivate all rules in Deactivate
```

Alternative injection procedure

```
InjectLiteralsIntoRules(Literals, Rules):
     for l in Literals:
          for r in Rules:
               if r.head equals l then remove r
               if r.tail contains l then remove l from r.tail
               if (!(Literals contains ~l) and r.tail contains ~l) then set r as not active
```

** optimization considerations**

Strict rules fixed point procedure

```
seekStrictRulesFixpoint():
    loop:
        Literals <- 0
        for s in Theory.unconsumedStrictRules:
            if empty(s.tail) then
                add s.head to Literals
                set s as not active
        if ~empty(Literals) then
            InjectLiteralsIntoRules(Literals, Theory.unconsumedRules)
            add Literals to +Delta
        else end loop
    return
```

Defeasible rules fixed point procedure

```
seekDefeasibleRulesFixpoint():
     loop:
          Irrefutables <- 0
          for r in Theory.unconsumedRules:
               if empty(r.tail) then
                    if CheckIrrefutability(r, Theory.unconsumedRules) then
                         add r.head to Irrefutables
                         set r as not active
                    else updateTmpPartial(r.head, r)
          if ~empty(Irrefutables) then
               InjectLiteralsIntoRules(Irrefutables, Theory.unconsumedRules)
               add Irrefutables to +delta
          else end loop
     return
```

Irrefutability check

```
checkIrrefutability(rule, Rules):
     for r in Rules:
          if r.head equals ~rule.head then
               if (Theory.supRelations contains (rule, r)
                    or Theory.supRelations ~contains (r, rule)) then return false
     return true
```

Temporary literals set update procedure

```
updateTmpPartial(literal, rule):
     for p in TmpPar:
          if p.literal equals ~literal then
               if Theory.supRelations contains (p.rule, rule) then
                    if rule prevails then
                         add (rule,literal) to TmpPar
                         remove p from TmpPar
                    else return
               else
                    remove p from TmpPar
                    return
     add (rule,literal) to TmpPar
```

Open problems

- Cycle elimination
- Do strict rules that are converted to defeasible rules have a higher priority?
- Do we have to remove those rules that have a defeasibly-derived literal as their head?
- Is it necessary to convert strict rules into defeasible rules even if we use the whole set of rules in the `seekDefeasibleRulesFixpoint` procedure?

Thank you for your attention!