ngames/interpreter.pl

query(*?Term*) *[det]*

Substitute for the usual `call/1` predicate, but includes support for terms expressed as conjunctions (e.g. A and B).

Arguments

Term	A Prolog term to be queried.

query_rule(*?Rule*) *[det]*

Return True if *Rule* is active given the current state of the system.

Arguments

Rule	A `rule/4` statement.

find_consequences(*+ID, +Type, +Threshold, -L*) *[det]*

Find the instantitations of the if-then-where rules of the *Type* kind that are currently active, and extract their consequences paired with their priority. Active rules with priority larger than *Threshold* are excluded.

Arguments

ID	Identifier for the action situation.
Type	One of either *boundary*, *position*, *choice* or *control*.
Threshold	The maximum priority of rules to be considered.
L	The output list with the processed consequences.

delete_key_gt(*L, N, NewL*) *[det]*

Auxiliary predicate to delete consequences whose priority is over some given threshold.

Arguments

L	List with priority-fluents pairs to be processed.
N	Integer threshold, fluents with priority over it are to be excluded.
NewL	List identical to *L*, but fluents with priority over Threshold are excluded.

process_consequences(*+Conseqs, +OldParts, ?NewParts*) *[det]*

Get all the consequences of some rule type and process them in decreasing order of priority. It returns a new list indicating the consequences that hold, negated ones (aka overwritten) included. The list of consequences is processed in a way such that consequences that are in conflict with consequences of higher priority are discarded.

It should be called as:

```
?- find_consequences(boundary,L),process_consequences(L,[],P).
```

Arguments

Conseqs	List of consequence of some rule type.
OldParts	List of old consequences. Intended to be called with the empty list.
NewParts	List of the consequences that hold, negations included.

get_simple_consequences(*+ID, +Type, +Threshold, -L*) *[det]*

Process the consequences of boundary, position, choice and payoff rules. It gets the consequences of the rules with the given identification and type, has rules of higher priority overwrite

rules of lower priority, and finally deletes negated (aka overwritten) facts. It returns the result in a list of fluents.

Arguments

ID	Identifier of the action situation.
Type	One of either *boundary*, *position*, *choice* or *payoff*.
Threshold	Consequences of rules with priorities exceeding *Threshold* are excluded.

control_conseq_fact_incompatible(+*Fact, +S*) *[det]*

Check whether a single fact is compatible with a list of established facts.

Arguments

Fact	The fluent whose compatibility we want to check.
S	List of previously established facts.

control_conseq_incompatible(+*Facts, +S*) *[det]*

Checks whether the fluents in *Facts* that make up a joint consequence statement of an active control rule are incompatible with the next states already derived in *S*.

Arguments

Facts	*Facts* derived from a new control rule. Either a single fact or a conjunction of them (aka *A and B*).
S	List of potential next state descriptions already derived (aka a list of lists).

control_rule_incompatible(+*Conseqs, +S*) *[det]*

Check whether the *Conseqs* list of an active control rule is compatible with the next states already derived in *S*.

Arguments

Conseqs	List of facts derived from a control rule. Each of them is either a single fact or a conjunction (aka *A and B*).
S	List of potential next state descriptions already derived (aka a list of lists).

add_rule_conseqs_to_next_states(+*Conseqs, +S, +P, +OldNextS, +OldNextP, -NewNextS, -NewNextP*)*[det]*

Given the potential next states derived in *S* and their probabilities in *P*, add the consequences in *Conseqs* of an active control rule.

Intended to be called as:

```
?- add_rule_conseqs_to_next_states(Conseqs,S,P,[],[],NewS,NewP).
```

Arguments

Conseqs	A list of consequences from an active control rule, with the format *Facts withProb P*.
S	List of alredy derived next states.
P	List of the probabilities of the already derived next states.
OldNextStates	List of partially processed next states.
OldNextP	List of partially processed next states probabilities.
NewNextS	List of next states after processing all of *Conseqs*.
NewNextP	List of next states probabilities after processing all of *Conseqs*.

add_joint_conseqs_to_next_states(*+F, +OldNextStates, +OldProb, -NewNextStates, -NewProb*)*[det]*

Given a fact (or conjunction of facts) alonside with their probability (*C withProb P*), derived from an activated control rule, update the list of *OldNextStates* and their probabilities *OldProb* into *NewNextStates* and *NewProb*.

Arguments

F	Consequence derived from an active control rule, joint by operator *withProb* to its probability.
OldNextStates	List of next states prior to update.
OldProb	List of next states probabilities prior to update.
NewNextStates	List of updated next states.
NewProb	List of updated next states probabilities.

add_joint_conseqs_to_single_state(*+F, +State, +Prob, -NewState, -NewProb*) *[det]*

Takes one consequence fact *F* (expressed as *C withProb P*) from an activated control rule, a state *S* to be updated (as a list of fluents) with probability *Prob* and return the updated state fluents and probability.

Arguments

F	A fact (or conjunction of facts) to append to the partially derived state *S*.
State	List of facts that make up a partially derived state.
Prob	Intermediate probability of the partially derived state.
NewState	List of updated state fluents.
NewProb	Updated probability (i.e. product of intermediate state probability and probability of the derived facts).

joint_conseqs_to_list(*+Old, +F, -New*) *[det]*

Auxiliary predicate to append a fact or a conjunction of facts (aka *A and B*) to a list of fluent.

Arguments

Old	List where new fluents are to be appended.
F	Fact or conjunction of facts to be appended.
New	Updated list of facts.

drag_compatible_fact(*+PreTranFact, +PostTranState, -UpdatedPostTransState*) *[det]*

If compatible, update a provisional next state with a fluent from the pre- transition state.

Arguments

PreTranFact	A fluent that holds true in some pre-transition state.
PostTranState	Partially constructed post-transition state.
UpdatedPostTransState	Post transition state updated with *PreTranFact*.

drag_compatible_fact(*+PreTranFact, +PostTranState, -UpdatedPostTransState*) *[det]*

Update a partially constructed post-transition state *PostTranState* with the compatible facts from pre-transition state *PreTranFact*

Arguments

PreTranFact	List of pretransition state facts.
PostTranState	Partially constructed post-transition state, as a list of facts.
UpdatedPostTransState	Updated post-transition state.

update_all_new_states(+*PreTranState, +PostTranStates, -UpdatedPostTransStates*) *[det]*

Update all the potential next states in *PostTranStates* with the compatible facts in *PreTranState*.

Arguments

PreTranState	List of fluents in the pre-transition state.
PostTranStates	Partially constructed post-transition states (aka a list of lists).
UpdatedPostTransStates	Updated post-transition states.

add_control_rules(+*RuleConseqs, +OldNextS, +OldNextP, -NewNextS, -NewNextP*) *[det]*

Given the priority-consequences pairs of the activated control rules in *RuleConseqs*, append them to the provisional states in *OldNextS* with their unadapted probabilities in *OldNextP*. If the rule is compatible with the facts already established, add the consequences into *NewNextS* and update the probabilities into *NewNextP*.

Arguments

RuleConseqs	List of priority-facts consequences derived from the activated control rules.
OldNextS	List of unupdated post-transition states (aka a list of lists).
OldNextP	List of unupdated probabilities for the post-transition states.
NewNextS	List of updated post-transition states (aka a list of lists).
NewNextP	List of updated post-transition states probabilities.

get_control_consequences(+*ID, +Threshold, +PreTranState, -PostTranState, -Probs*) *[det]*

Gather the consequences of control rules given the current state for the action situation of interest. The rules whose priority exceeds *Threshold* are excluded.

Arguments

ID	Identifier of the action situation.
Threshold	Rules whose priority exceed it are excluded from processing.
PreTranState	List of pre-transition state facts.
PostTranState	List of possible post-transition states (aka a list of lists).
Probs	List of probabilities of the post-transition states.