uses/2

Description

```
uses(Object, Predicates)
uses(Object, PredicatesAndAliases)
```

Declares that all calls (made from predicates defined in the category or object containing the directive) to the specified predicates are to be interpreted as messages to the specified object. Thus, this directive may be used to simplify writing of predicate definitions by allowing the programmer to omit the `Object::` prefix when using the predicates listed in the directive (as long as the predicate calls do not occur as arguments for user-defined meta_predicates or for non-standard Prolog meta_predicates not declared on the config files).

It is possible to specify a predicate alias using the notation `Functor/Arity::Alias/Arity`. Aliases may be used either for avoiding conflicts between predicates specificed in several `uses/2` directives or for giving more meanfingul names considering the using context of the predicates.

Template and modes

```
uses(+object_identifier, +predicate_indicator_list)
uses(+object_identifier, +predicate_indicator_alias_list)
```

Examples

```
:- uses(list,
    [append/3, member/2]).

foo :-
    ...,
    findall(X, member(X, L), A),    % the same as: findall(X, list::member(X, L), A)
    append(A, B, C),                % the same as: list::append(A, B, C)
    ...
```

Another example, using the extended notation that allows us to define predicate aliases:

```
:- uses(btrees, [new/1::new_btree/1]).
:- uses(queues, [new/1::new_queue/1]).

btree_to_queue :-
    ...,
    new_btree(Tree),     % the same as: btrees::new(Tree)
    new_queue(Queue),    % the same as: queues::new(Queue)
    ...
```


Last updated on: October 26, 2005