Expanding prefixed names

Joop Ringelberg 07-04-20 Version: 1

Introduction

The syntax of the Perspectives Language describes a construct `prefixedName` (see *Syntax of the Perspectives Language*). An example of a prefixed name, taken from `model:System`, is `sys:PerspectivesSystem`. The prefix `sys:` expands into `model:System$`, which gives us `model:System$PerspectivesSystem`. However, not all positions that allow identifiers support this expansion. This text gives the details.

Declaring prefixes

Prefixes are local to the model in which they are used. Even stronger, they are local to the *context* in which they are declared (where `domain` is a context, too). The modeller declares a prefix with a line like this:

```
use: sys for model:System
```

(notice we omit the ':' and '$' for clarity).

Prefix declarations shadow each other; if we were to declare a prefix `sys` local to, for example, `sys:TrustedCluster`, binding it to `model:SomeOtherNamespace`, this means that an identifier prefixed with sys nested inside `sys:TrustedCluster` will expand to `model:SomeOtherNamespace`.

Syntactic locations that support prefix expansion

Some syntactic locations where an identifier is expected, support prefix expansion. These are the locations listed below (enclosed in < and > in the examples). In these situations we must use a **segmented name**, such as `PerspectivesSystem$User` (but notice that a single segment such as `User` is ok).

1. `aspect <aspectName>.`
2. `indexed <indexName>.`
3. `callExternal <functionName>.`
4. `callEffect <effectName> returns <resultType>.`
5. `filledBy: <roleName>.`
6. `view: (<propertyName>).`
7. all references to views:
    a. `perspective on: SomeRole (<AView>)`. Here, <AView> is the default view.
    b. `Change with <AnotherView>`. <AnotherView> is a verb-specific view.

    c. `indirectObject SomRole (<IndirectObjectView>)`. The view specific to the indirect object of an action.
8. All occurrences of role- or property names in query expressions.
9. All occurrences of *indexed names* in query expressions.

Syntactic locations that don't support prefix expansion

We also have syntactic locations where we must use a **local name** (i.e. a capitalised name: the production `segment` in the identifier grammar):

1. All declared names:
    a. all context kinds (`domain, case, party, activity, state`), e.g. `case <caseName>`
    b. all role kinds (`thing, context, user, external`), e.g. `role <roleName>`
    c. properties: `property <propertyName>`
    d. views: `view <viewName>.`
2. `perspective on: <object>`
3. `indirectObject: <object>`
4. `bot: for <user>`

All these local names are automatically scoped to their enclosing namespace. This means that, for example for a role, the name of the context is tacked on front.

Implementation notices

A model source text is parsed using the functions in the module `Perspectives.Parsing.Arc`. Query expressions within the text are parsed using the functions in the module `Perspectives.Parsing.Arc.Expression`. Neither of these modules expand prefixes.

Instead, this is done exclusively in the modules

- `Perspectives.Parsing.Arc.PhaseTwo`;
- `Perspectives.Query.ExpandPrefix`,

for, respectively, identifiers in the main text and in query expressions.

A model source text is compiled to a `DomeinFile` of type data. The last stage of this compilation is performed by the module `Perspectives.Parsing.Arc.PhaseThree`. We can safely assume that the data structures traversed by functions in this module contain just expanded identifiers; prefixed names do no longer occur in them.