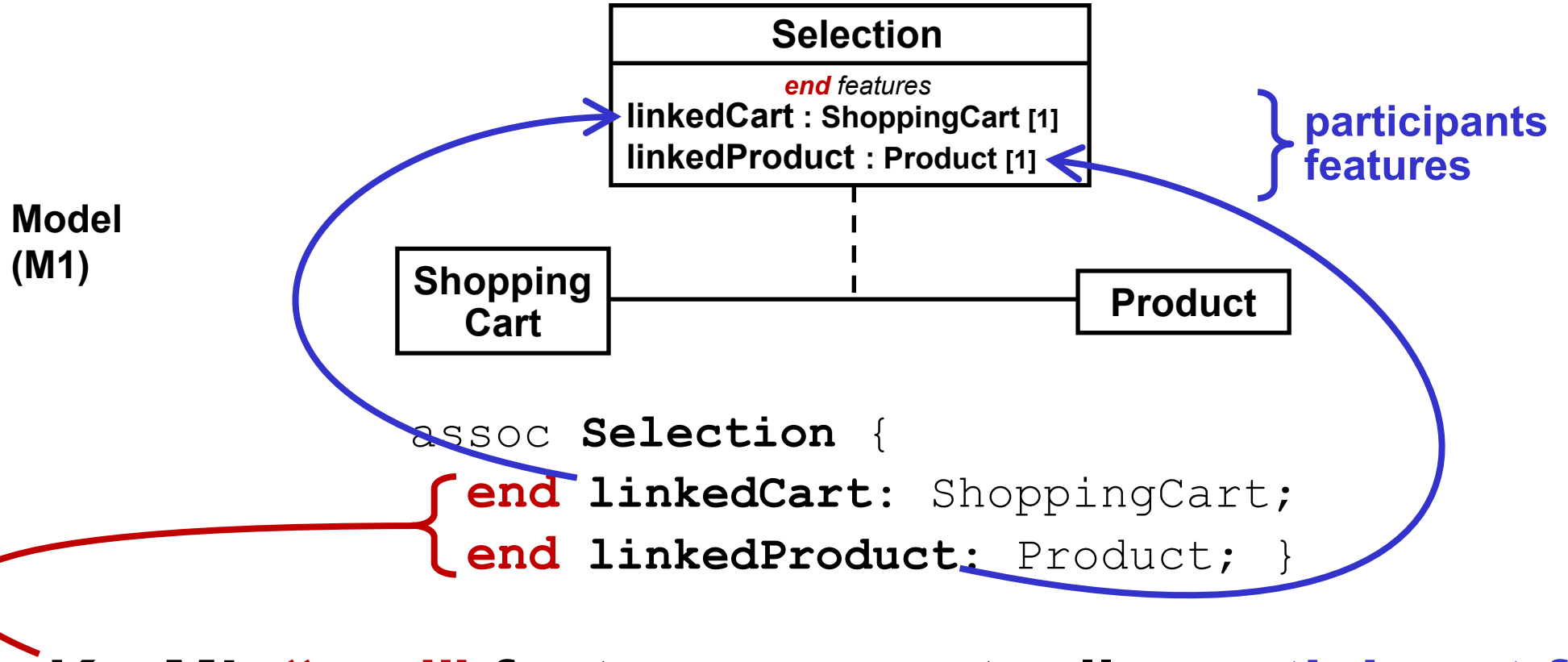
Overview

- Associations
- Debt (SysML1 & SST)
  - Paid (SST February)
  - **Unresolved (still paying interest)**
- Proposals
- Summary

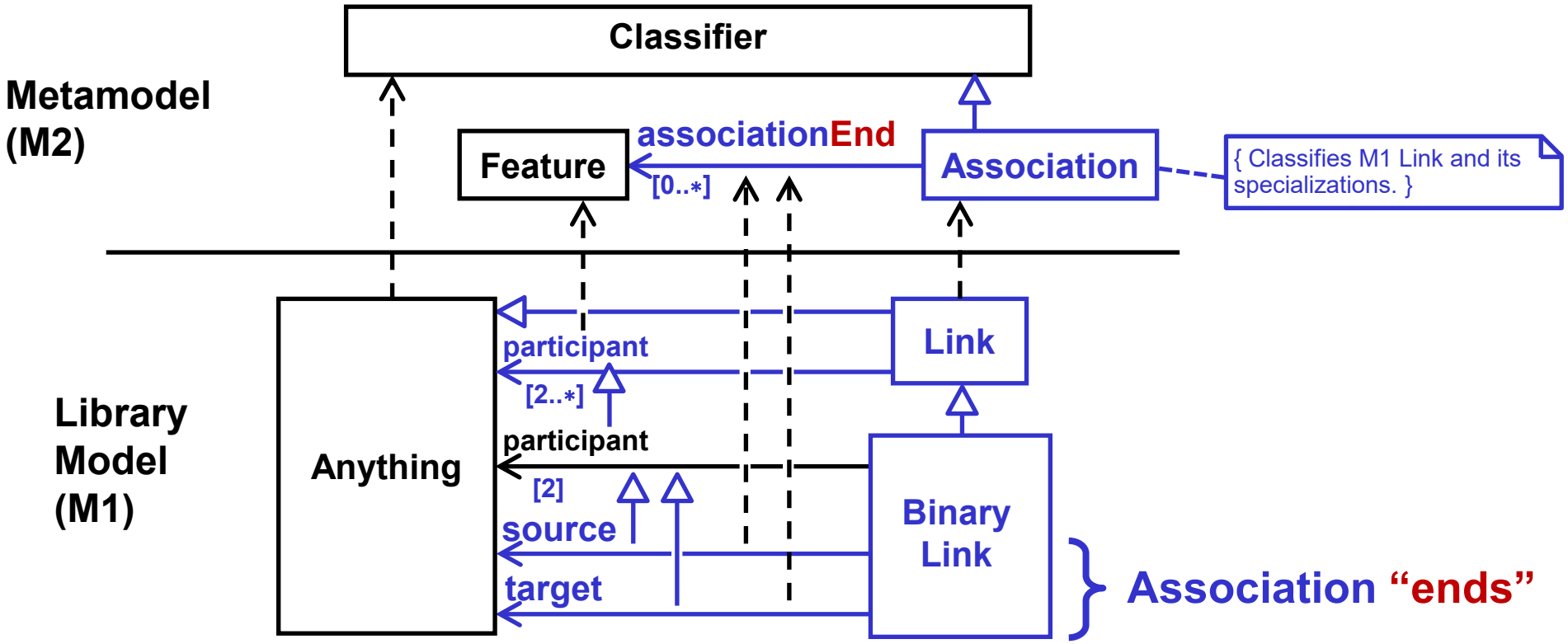
Assoc "Ends" (textual syntax)



- KerML "end" features are actually participant features.
  - Values are the things being linked, exactly one each.
  - Better term? Needs to be short, not an abbreviation.

Assoc “Ends” (abstract syntax)

KERML-34

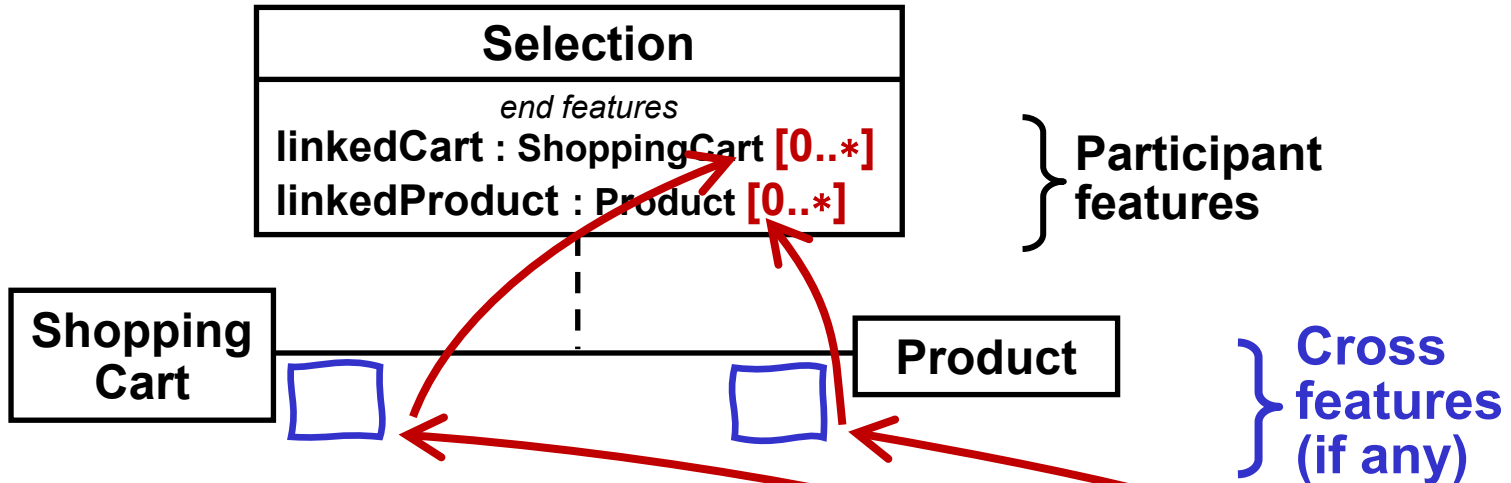


- **Same term in the metamodel.**

Assoc Textual Syntax (multiplicity)

KERML-26

Model (M1)




```
assoc Selection {
  end linkedCart: ShoppingCart[0..*];
  end linkedProduct: Product[0..*];}
```

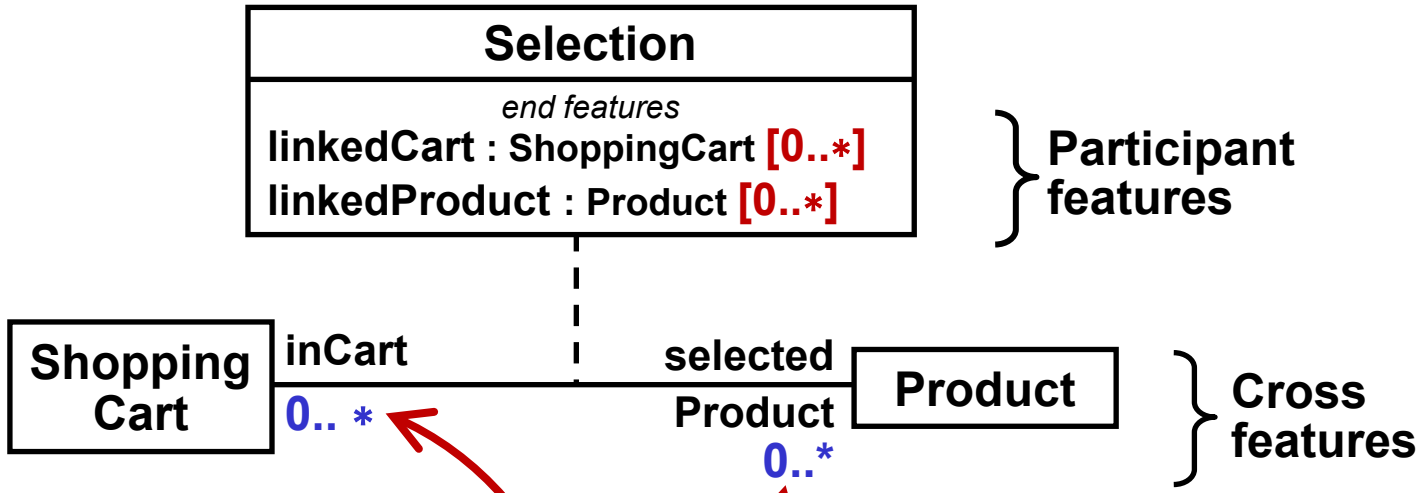
- "End" multiplicity is about **cross features** …,
- … which might not exist.
  - Useful for one-way and non-navigable associations.
- Looks like participant multiplicity (if you know "end").

Same for ordering and uniqueness

Assoc Textual Syntax (multiplicity)

KERML-36

Model (M1)

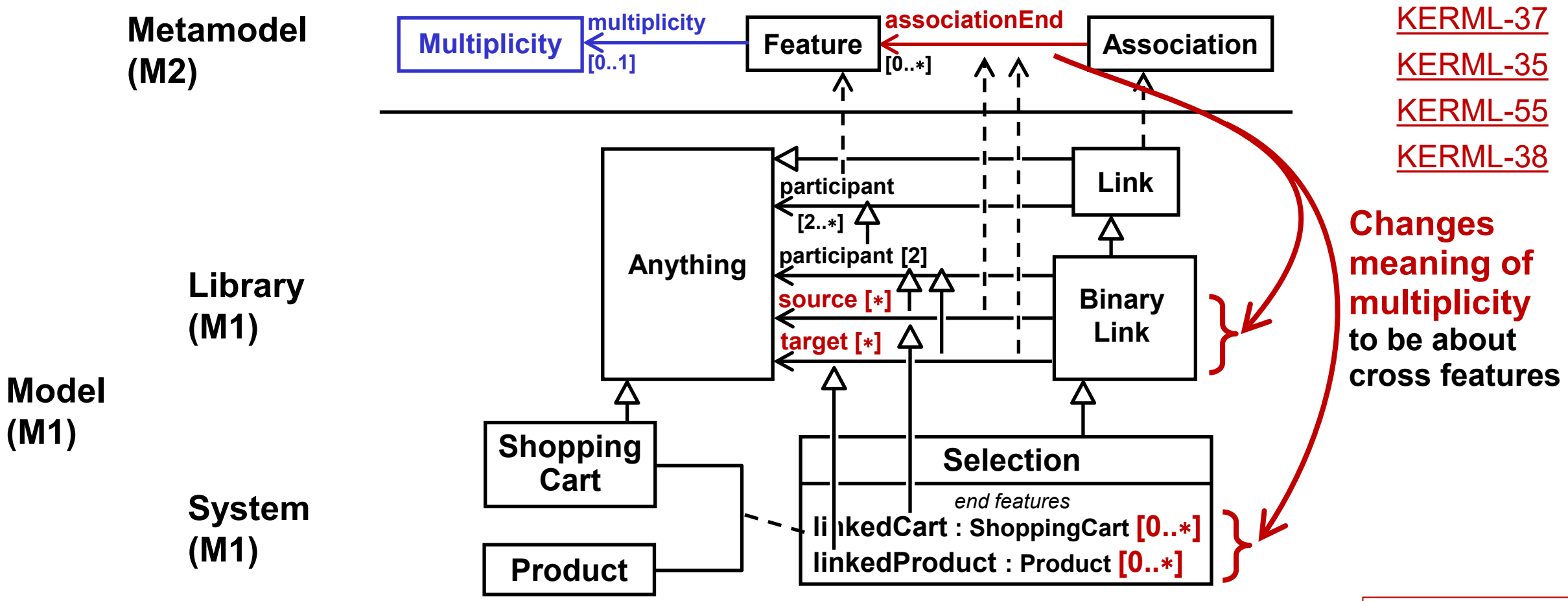



```
assoc Selection {
  end linkedCart: ShoppingCart[0..*] subsets linkedProduct.inCart;
  end linkedProduct: Product[0..*] subsets linkedCart.selectedProduct;}
```

- "End" multiplicity **redundant** and possibly **conflicting**
  - between "end" and cross features.
- In textual **and** abstract syntax.

Same for ordering and uniqueness

Assoc Abstract Syntax (multiplicity)

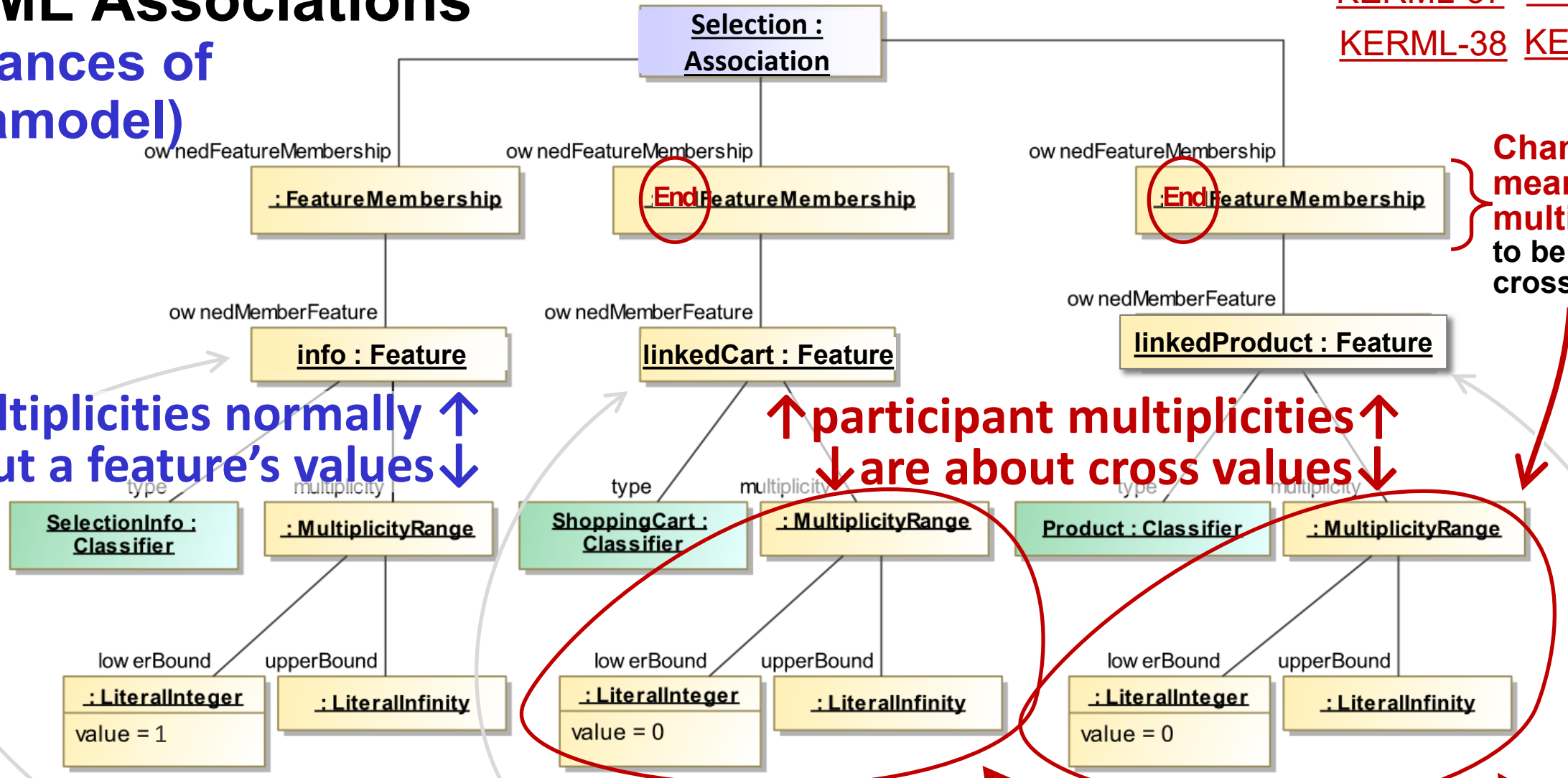


- **"End" changes feature multiplicity semantics …**
  - **… to be about cross features.**
  - **Number of participant feature values not restricted.**

Same for ordering and uniqueness

KerML Associations
(instances of metamodel)

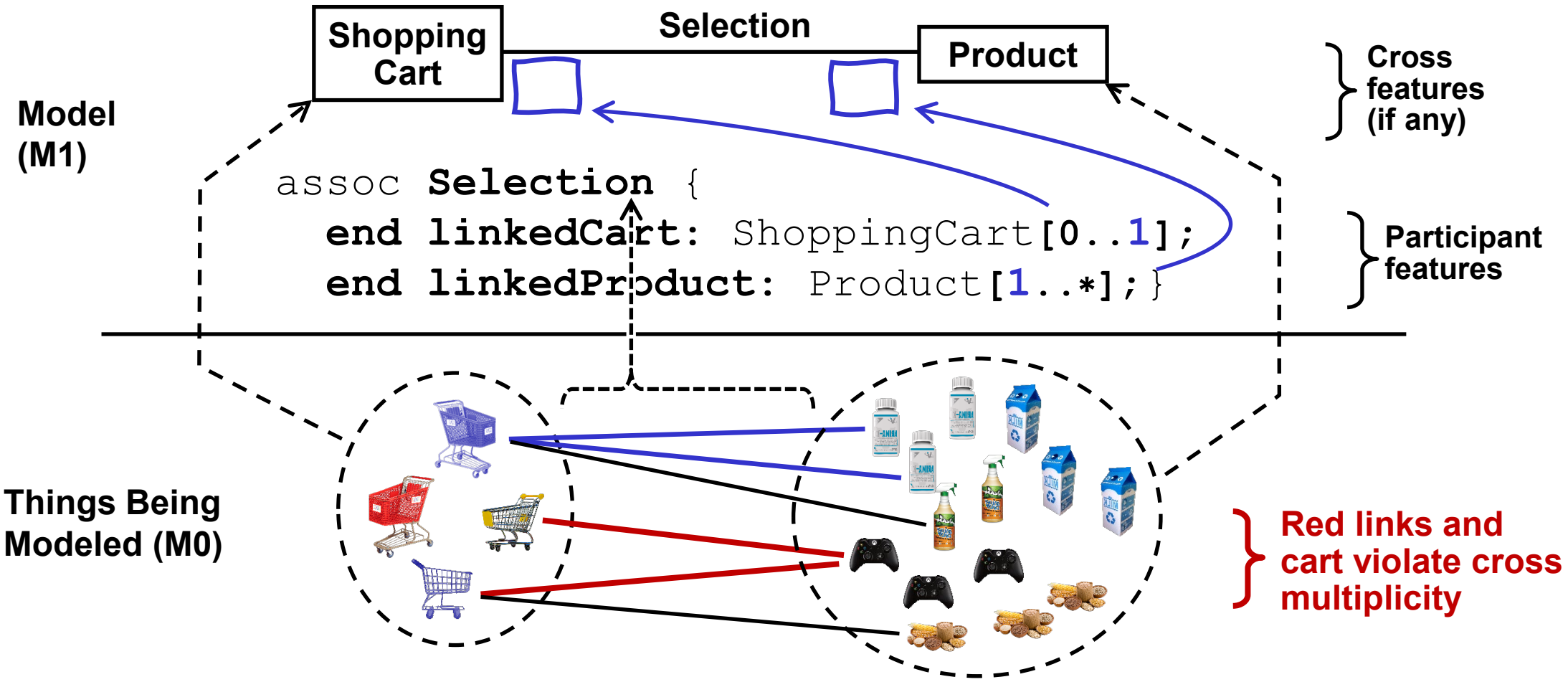
KERML-37 KERML-35
KERML-38 KERML-55




```
assoc Selection {
  end cart: ShoppingCart[0..*];
  end selectedProduct: Product[0..*];
  feature info: SelectionInfo [1..*]; }
```

Same for ordering and uniqueness

Assoc Semantics (cross multiplicity)



- **Every cart and product must satisfy cross multiplicity.**
  - **via Selection links.**

Assoc Semantics (cross multiplicity)

KERML-40

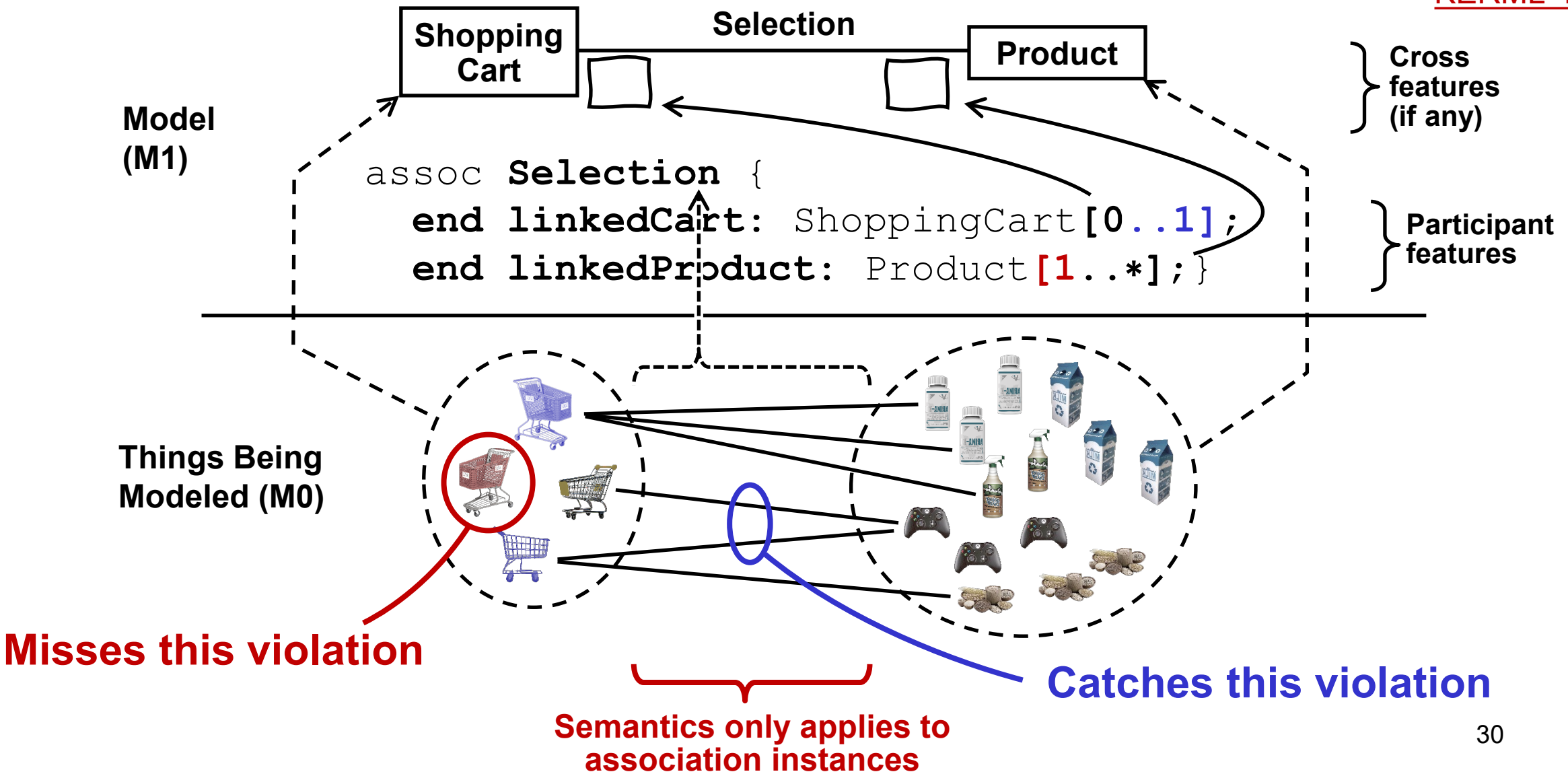
- **Informal text currently.**
- **Applies only to instances of association (links),**
  - **rather than instances of the associated classifiers.**

[7.4.5] if an association end has a multiplicity specified other than 1..1, then this is interpreted as follows: *For each association end, the multiplicity, ordering and uniqueness constraints specified for that end apply to each set of instance of the association that have the same (single) values for each of the other ends.* For a binary association, this corresponds to the multiplicity resulting from "navigating" across the association given a value at one end of the association to the other end of the association.

[8.4.4.5.1] If an associationEnd has a declared multiplicity other than 1..1, then this shall be interpreted as follows: For an Association with *N* associationEnds, consider the *i*-th associationEnd e_i. *The multiplicity, ordering and uniqueness constraints specified for* e_i *apply to each set of instances of the Association that have the same (singleton) values for each of the N-1 associationEnds other than* e_i.

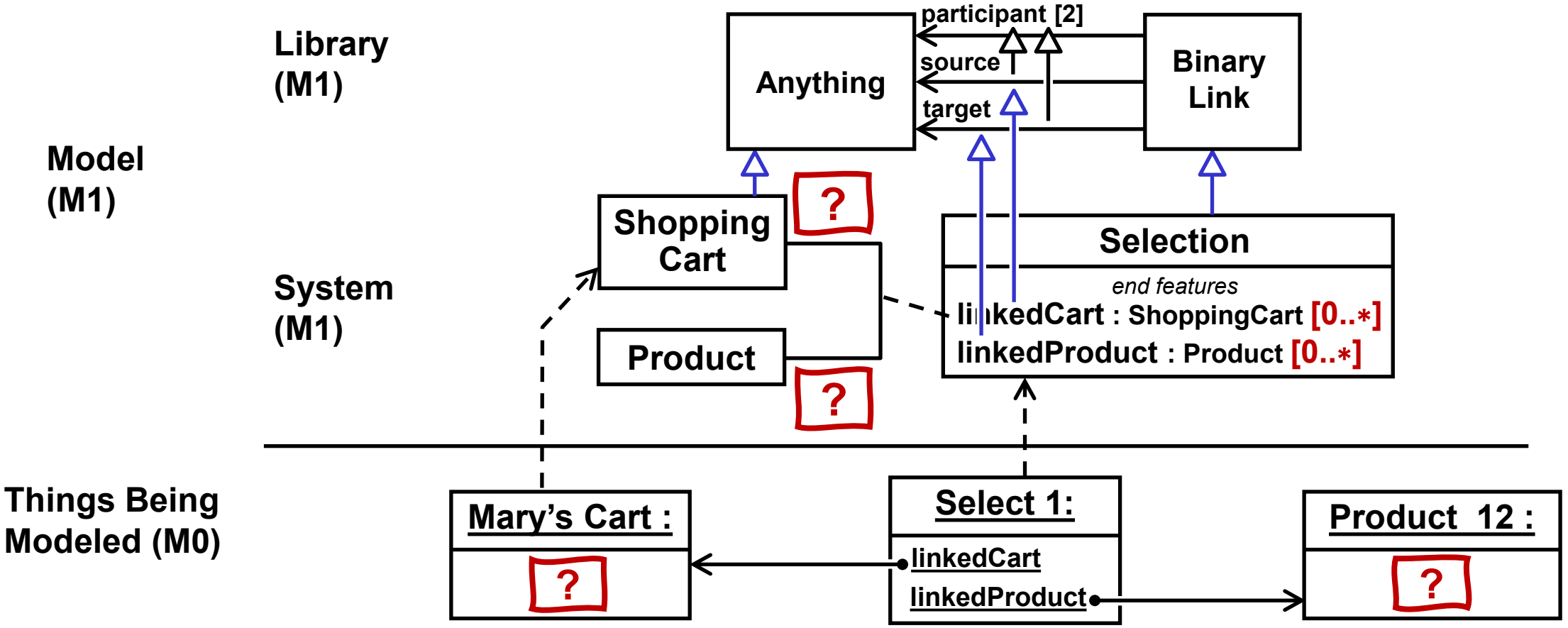
Assoc Semantics (cross multiplicity)

KERML-40



Assoc, Identifying Cross Features

KERML-41



- Nothing in the textual or abstract syntax for them.
  - Cross-subsetting chain pattern is modeler option.
  - API access can't depend on it, eg, for automated analysis.

Association Debt Summary

1. **Misleading textual syntax**
   - Term: "end" for participant rather than cross features KERML-34
   - Cross feature multiplicity looks like participant's. KERML-26
2. **Two meanings for feature multiplicity** KERML-37 KERML-55 KERML-38
   - Requires special casing in modeling/analysis tools.
   - Redundant cross feature multiplicity KERML-36
3. **Incomplete semantics** KERML-35
   - Number of participant values not restricted (should be [1]).
   - Cross multiplicity semantics is informal and incomplete. KERML-40
4. **Cross features not identified** KERML-41
   - Subsetting pattern is non-standard (tool builders and modelers cannot depend on it).

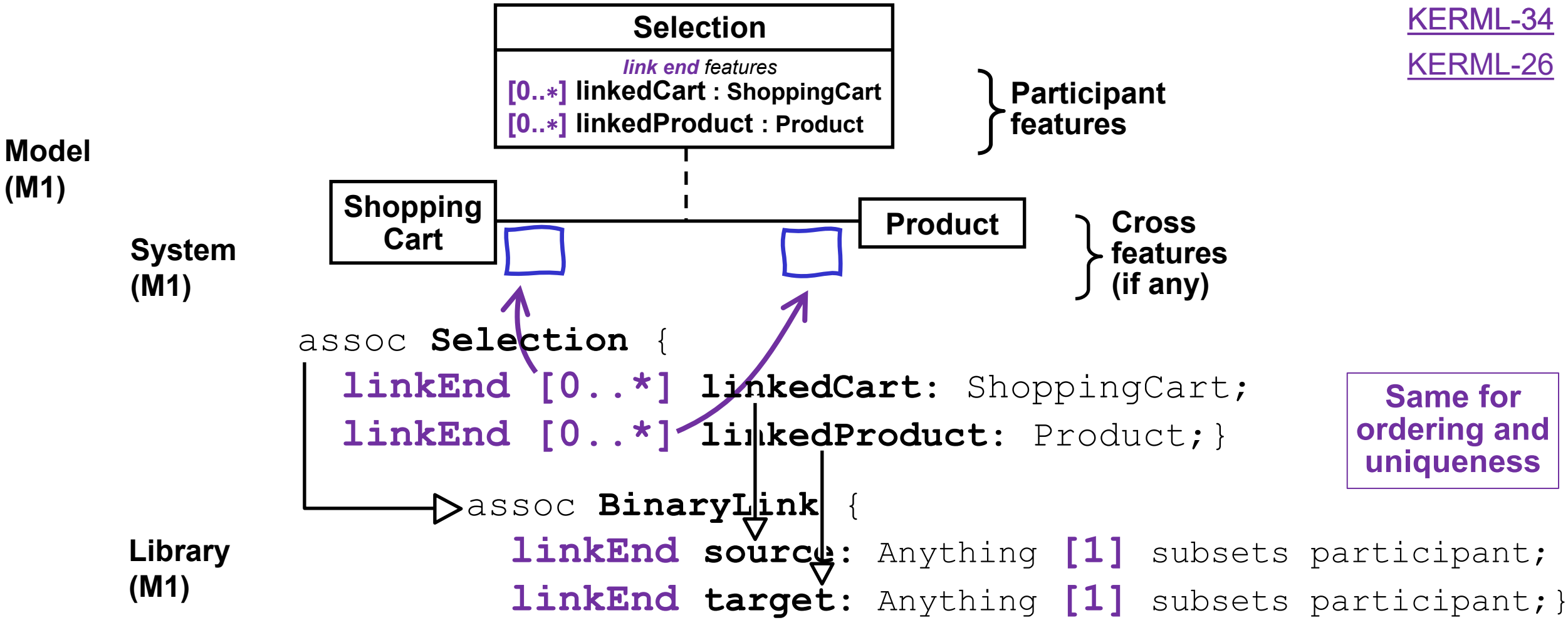
Overview

- Associations
- Debt (SysML1 & SST)
  - Paid (SST February)
  - Unresolved (still paying interest)
- **Proposals**
- Summary

Assoc Participant Textual Syntax (Proposal)

KERML-34

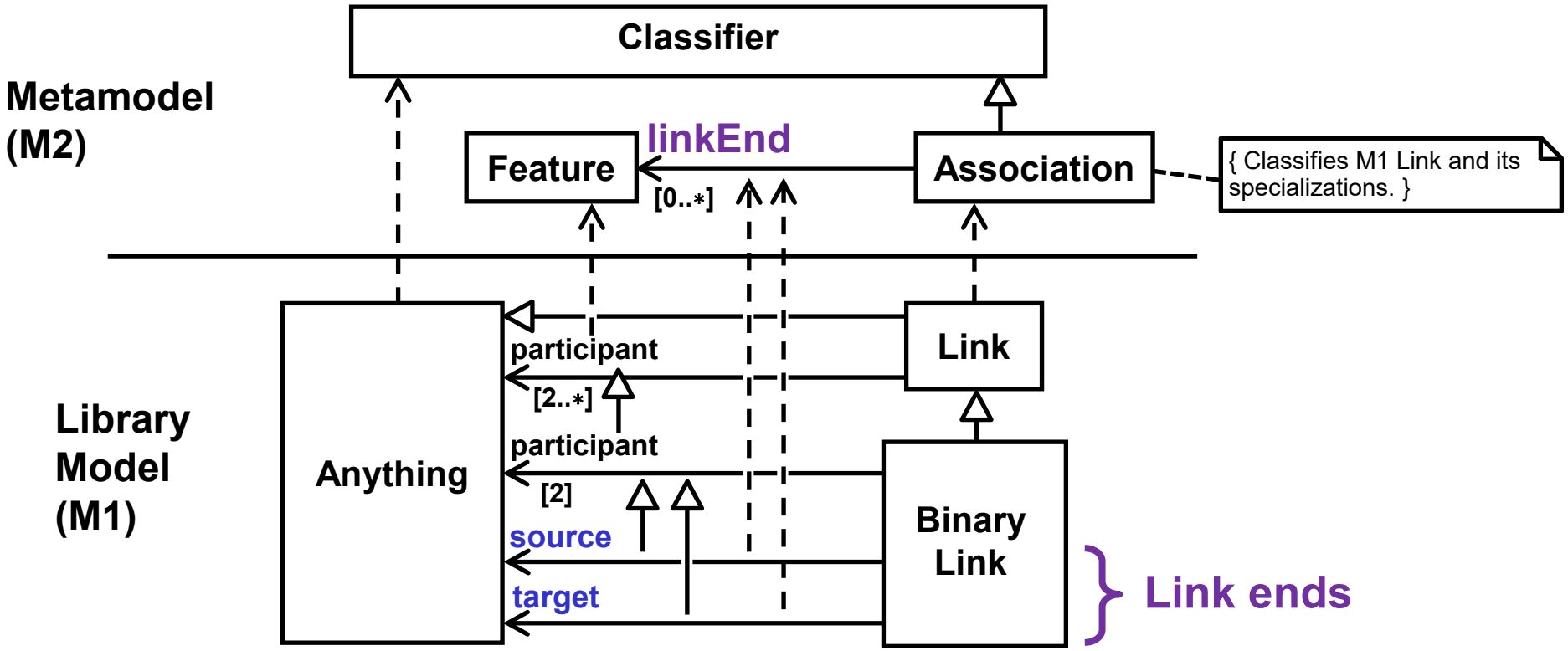
KERML-26



- **Keyword and cross multiplicity position change.**
  - **Leaves (textual) room for participant multiplicity in library.**

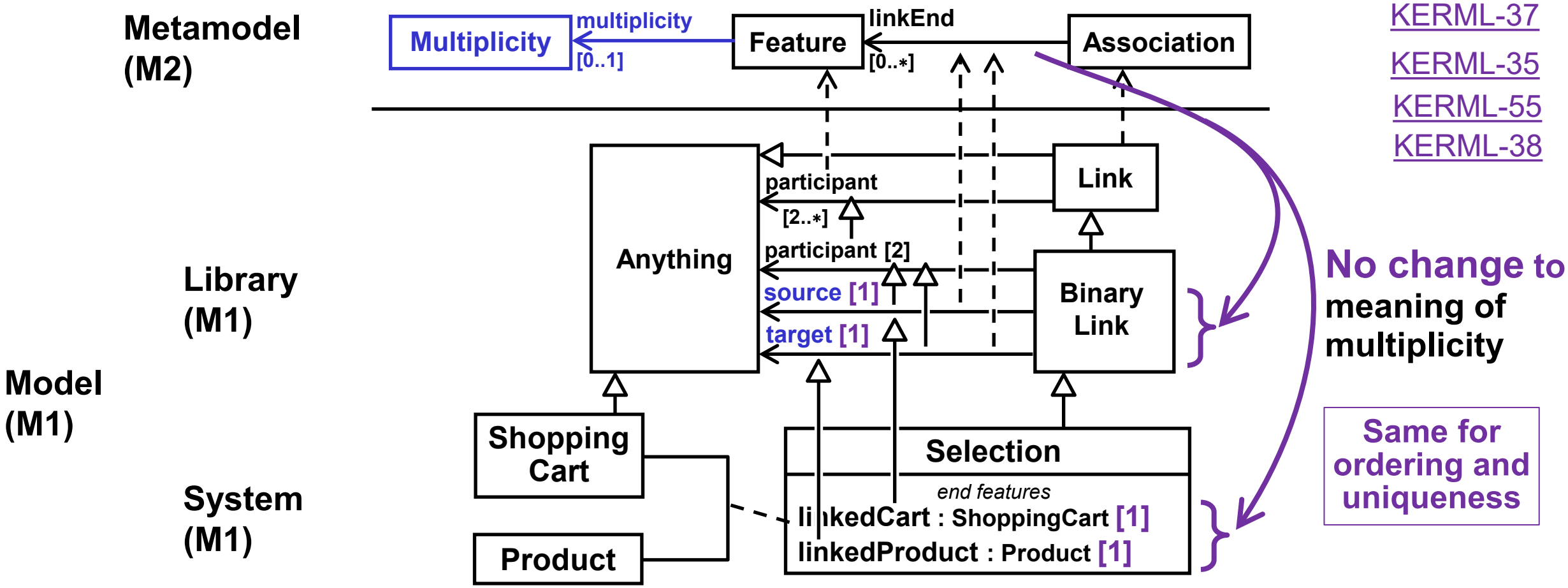
Assoc Participant Abstract Syntax (Proposal)

KERML-34



- **Same term in the metamodel.**

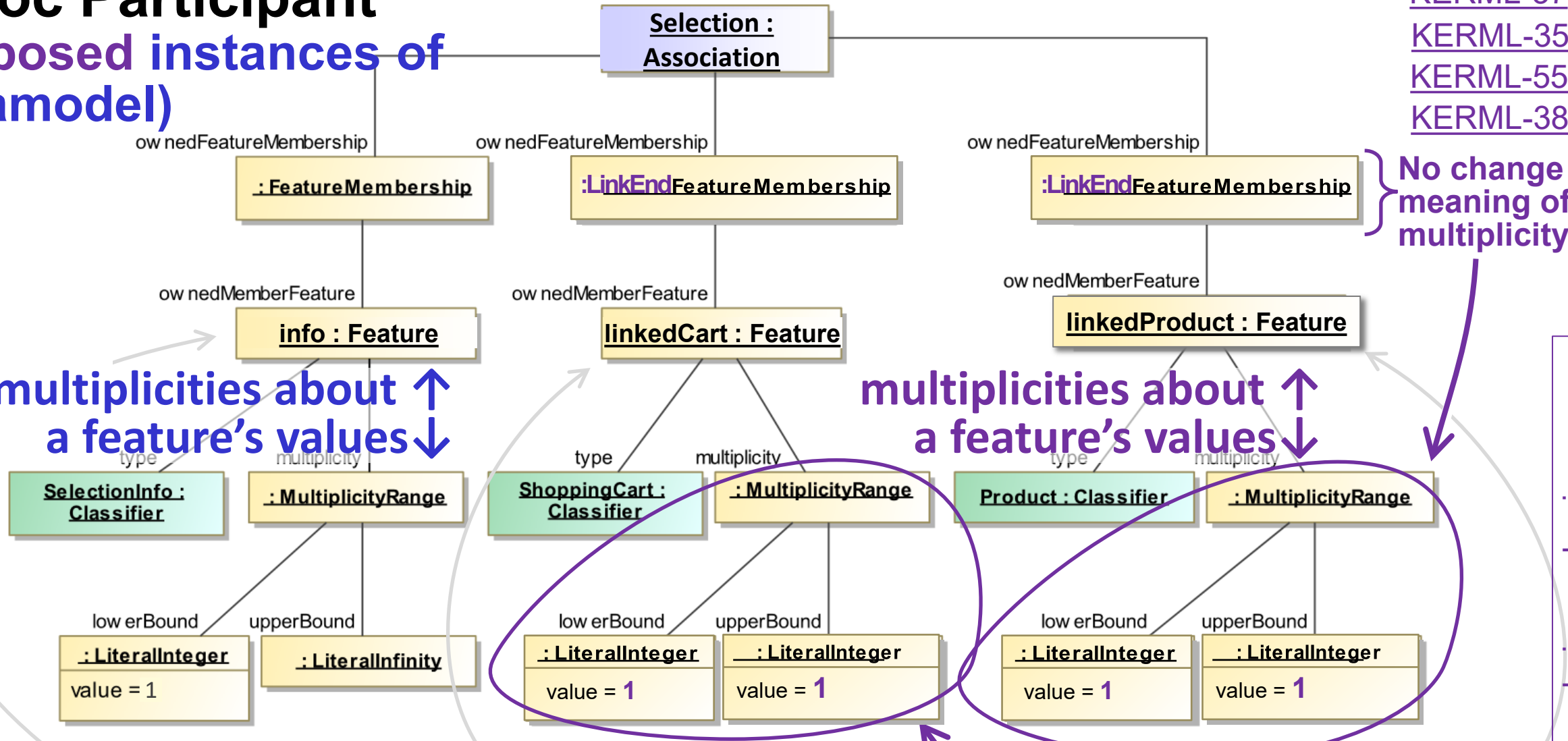
Assoc Participant Semantics (Proposal)



- No change in feature multiplicity semantics …
  - … to be about cross features.
  - Number of participant feature values restricted to 1.

Assoc Participant
(proposed instances of metamodel)

KERML-37
KERML-35
KERML-55
KERML-38

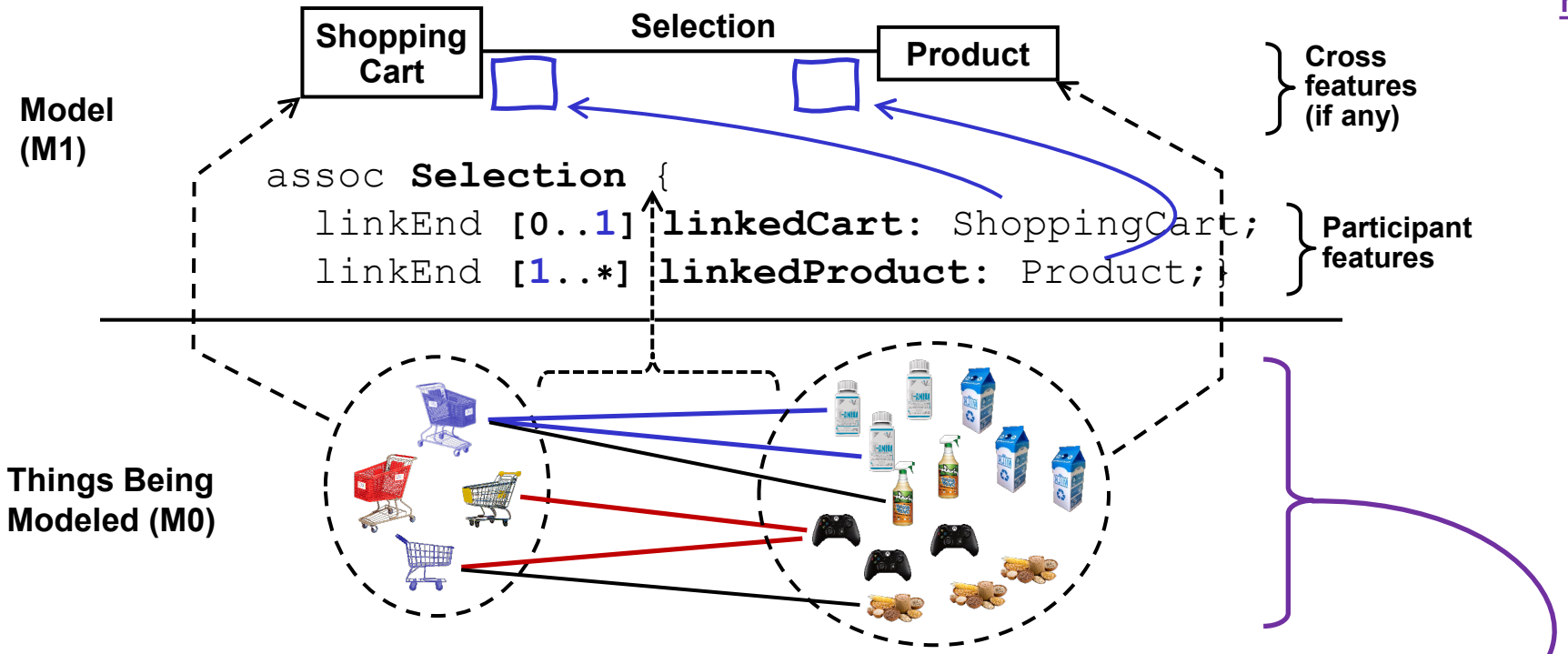



```
assoc Selection {
  end cart: ShoppingCart[1];
  end selectedProduct: Product[1];
  feature info: SelectionInfo [1..*]; }
```

Same for ordering and uniqueness

Assoc Cross Feature Semantics (How?)

KERML-40



- How constrain all instances collectively?
  - Of association and its associated classifiers
- Usually done in math semantics
  - Trying to use math only in Core.

Cross Feature Modeled Semantics (Proposal)

KERML-40

Model (M1)

Cross features if any

```
assoc Selection {
  linkEnd [0..1] linkedCart: ShoppingCart;
  linkEnd [1..*] linkedProduct: Product;}
```

Parses to

```
assoc Selection {
  linkEnd linkedCart: ShoppingCart
          subsets linkedProduct.(linkedCart::inCart);
  ⊕ feature inCart: ShoppingCart [0..1] featured by Product;

  linkEnd linkedProduct: Product
          subsets linkedCart.(linkedProduct::selectedProduct);
  ⊕ feature selectedProduct: Product [1..*] featured by ShoppingCart
}
```

Same for ordering and uniqueness

- **Cross features in participant namespace.**

Cross Feature ~~Redundancy~~ (Proposal)

KERML-36

Model (M1)

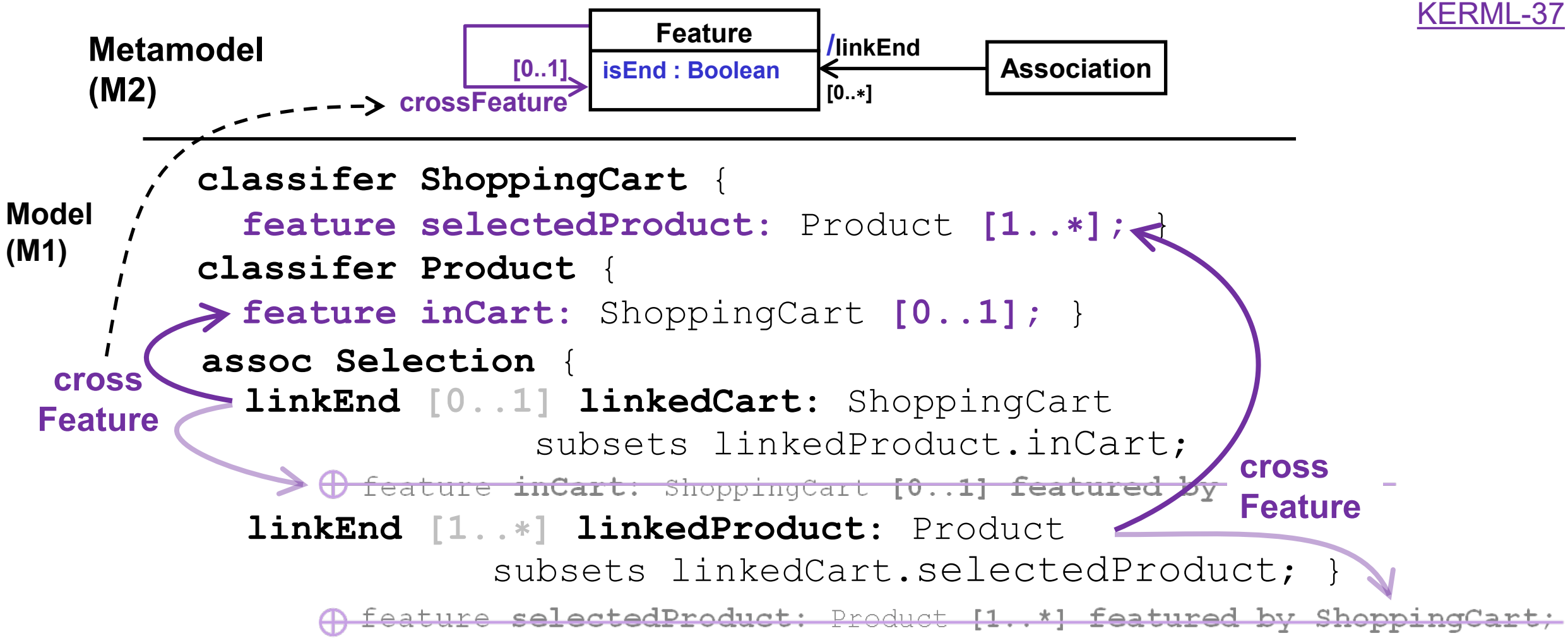
```
classifer ShoppingCart {
  feature selectedProduct: Product [1..*];}
classifer Product {
  feature inCart: ShoppingCart [0..1]; }
assoc Selection {
  linkEnd [0..1] linkedCart: ShoppingCart
                 subsets linkedProduct.inCart;
⊕ feature inCart: ShoppingCart [0..1] featured by Product;
  linkEnd [1..*] linkedProduct: Product
                 subsets linkedCart.selectedProduct; }
⊕ feature selectedProduct: Product [1..*] featured by ShoppingCart;
```

Same for ordering and uniqueness

- **Move cross features to associated classifiers**
  - **As UML/SysML tools currently do with "association-owned ends".**

Cross Feature Abstract Syntax (Proposal)

KERML-37



- **Participant (link end) features identify cross features**
  - **As UML/SysML associations currently do with assoc ends.**

Participant, Identify Cross Subsets (Proposal)

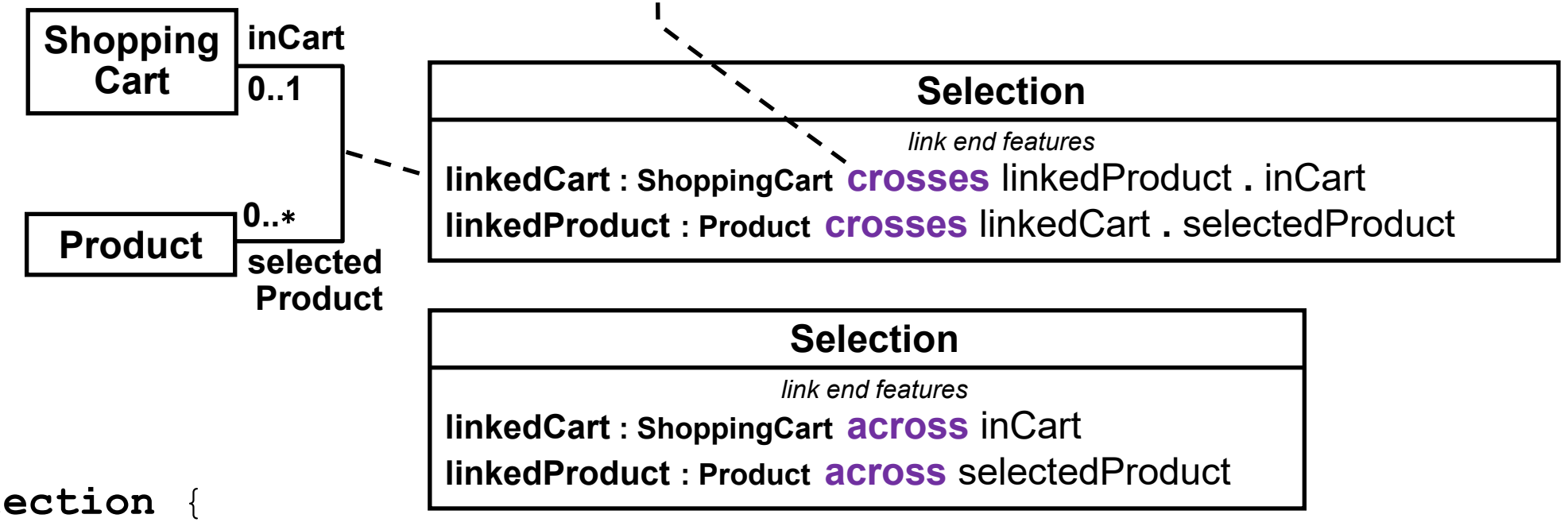
KERML-41

Metamodel (M2)

For binary associations:

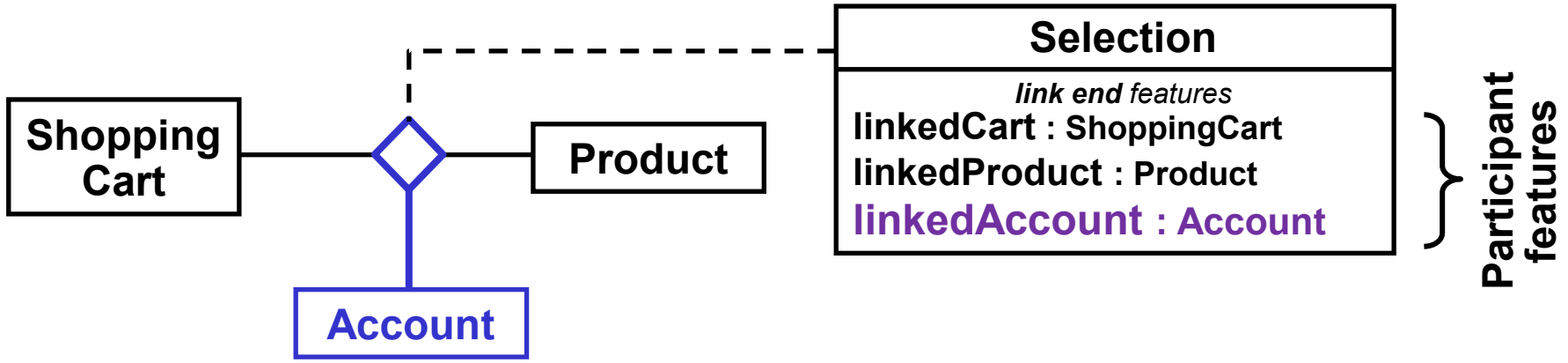
- **source must be link end feature that identifies a cross feature**
- **target must chain through other link end feature then cross feature**

Model (M1)




```
assoc Selection {
  linkEnd [0..*] linkedCart: ShoppingCart across inCart;
  linkEnd [0..1] linkedProduct: Product across selectedProduct;}
```

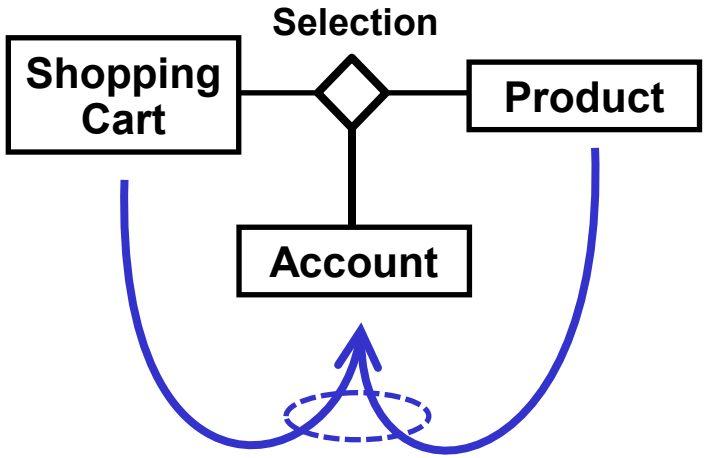
“N”-aries (> 2 participants)



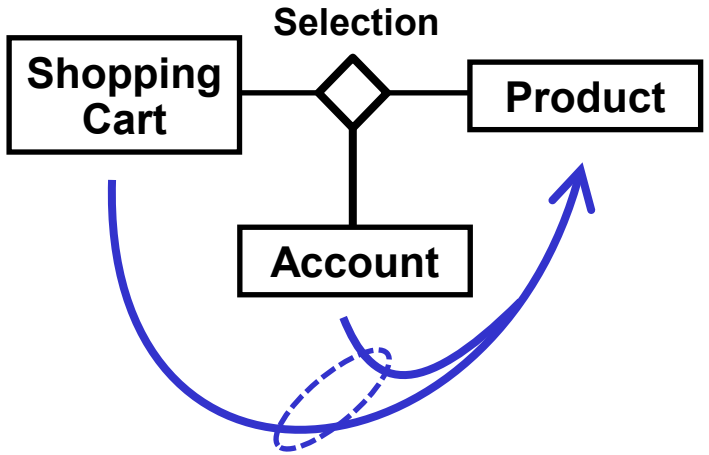

```
assoc Selection {
  linkEnd linkedCart: ShoppingCart [1] subsets participant;
  linkEnd linkedProduct: Product [1] subsets participant;
  linkEnd linkedAccount: Account [1] subsets participant;
  feature redefines participant: Anything [3] }

        assoc Link {
            feature participant: Anything [2..*]; }
```

- **Specialize Link & participant directly.**

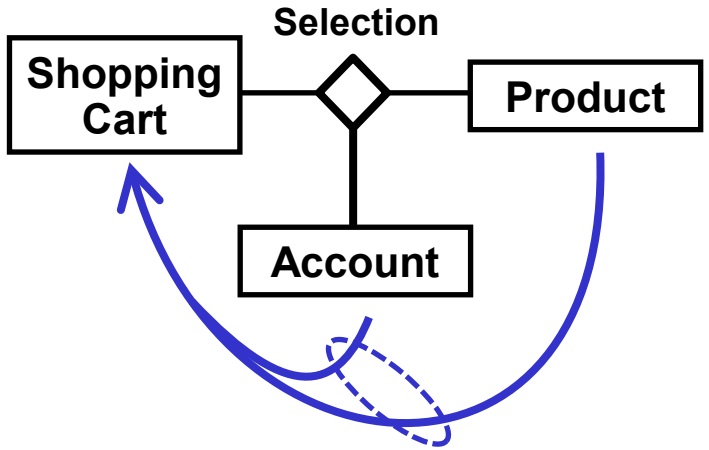
“N”-aries (> 2 participants) “Navigation”



a shopping cart + a product → account

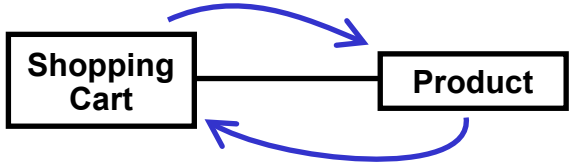


a shopping cart + an account → product(s)



a product + an account → shopping cart

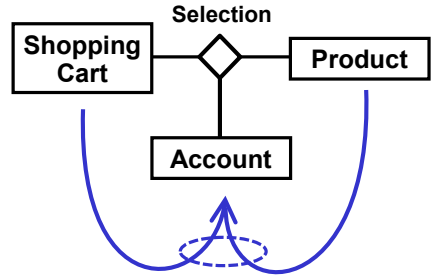
- **Association end (cross) multiplicity, ordering, and uniqueness apply to “navigation” from n-1 things.**
  - **Generalized from n=2.**

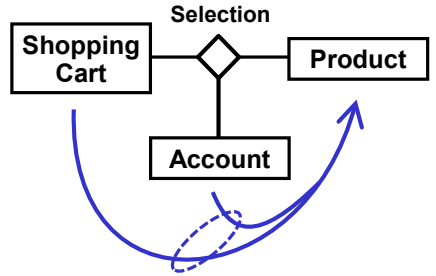


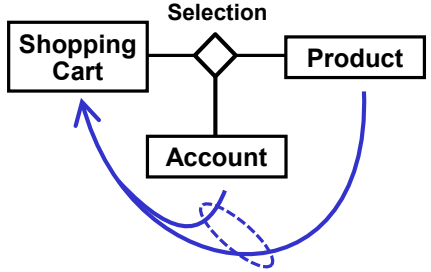
"N"-ary Cross Features (Proposal) KERML-40

- **Domains are sequences of "n-1" data types.**

```
datatype CartProductPair specializes OrderedPair {
  redefines element1: Cart;
  redefines element2: Product; }}

datatype CartAccountPair specializes OrderedPair {
  redefines element1: Cart;
  redefines element2: Account; }}

datatype ProductAccountPair specializes OrderedPair {
  redefines element1: Product;
  redefines element2: Account; }}
```



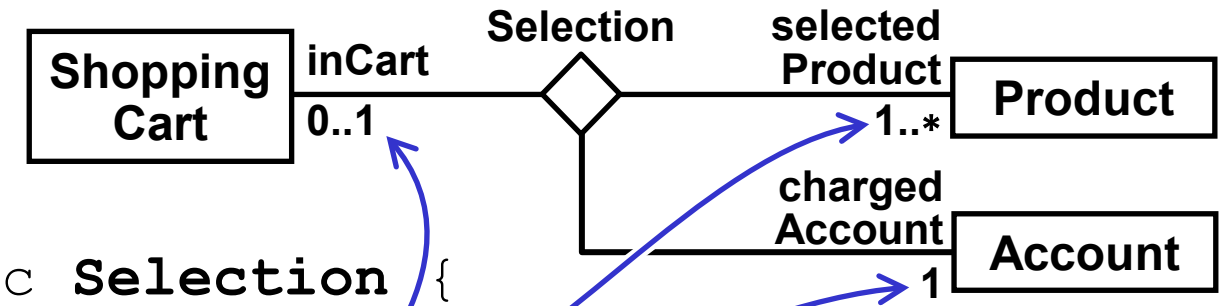





```
datatype OrderedPair specializes OrderedCollection {
  feature redefines elements [2];
  feature element1 [1] = elements#(1); feature element2 [1] = elements#(2);}
```

N-ary Assoc Modeled Semantics (Proposal)

KERML-40



Model (M1)

```
assoc Selection {
  linkEnd [0..1] linkedCart: ShoppingCart;
  linkEnd [1..*] linkedProduct: Product;
  linkEnd [1] linkedAccount: Account; }
```

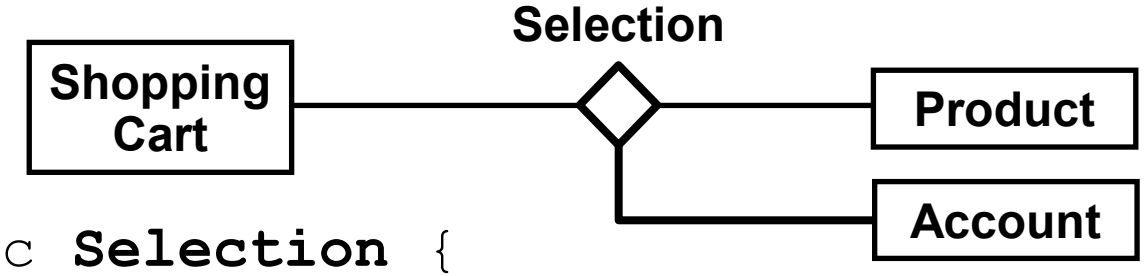
Parses to

```
assoc Selection {
  linkEnd linkedCart: ShoppingCart;
    ⊕ feature inCart: ShoppingCart [0..1] featured by ProductAccntPair;
  linkEnd linkedProduct: Product;
    ⊕ feature selectedProduct: Product [1..*] featured by CartAccntPair;
  linkEnd linkedAccount: Account;
    ⊕ feature chargedAccount: Account [1] featured by CartProductPair;
```

Same for ordering and uniqueness

N-ary Assoc Modeled Semantics (Proposal)

KERML-40



Model (M1)

```
assoc Selection {
    linkEnd [0..1] linkedCart: ShoppingCart;
    linkEnd [1..*] linkedProduct: Product;
    linkEnd [1] linkedBuyer: Account; }
```

Parses to

```
assoc Selection {
  linkEnd linkedCart: ShoppingCart
    subsets (linkedProduct, linkedAccount).(linkedCart::inCart);
      ⨁ feature inCart: ShoppingCart [0..1] featured by ProductAccntPair;
  linkEnd linkedProduct: Product
    subsets (linkedCart, linkedAccount).(linkedProduct::selectedProduct); }
      ⨁ feature selectedProduct: Product [1..*] featured by CartAccntPair;
  linkEnd linkedAccount: Account
    subsets (linkedCart, linkedProduct).(linkedBuyer::productBuyer); }
      ⨁ feature productBuyer: Account [1] featured by CartProductPair;
```

Same for ordering and uniqueness

N-ary, Identify Cross Subsets (Proposal)

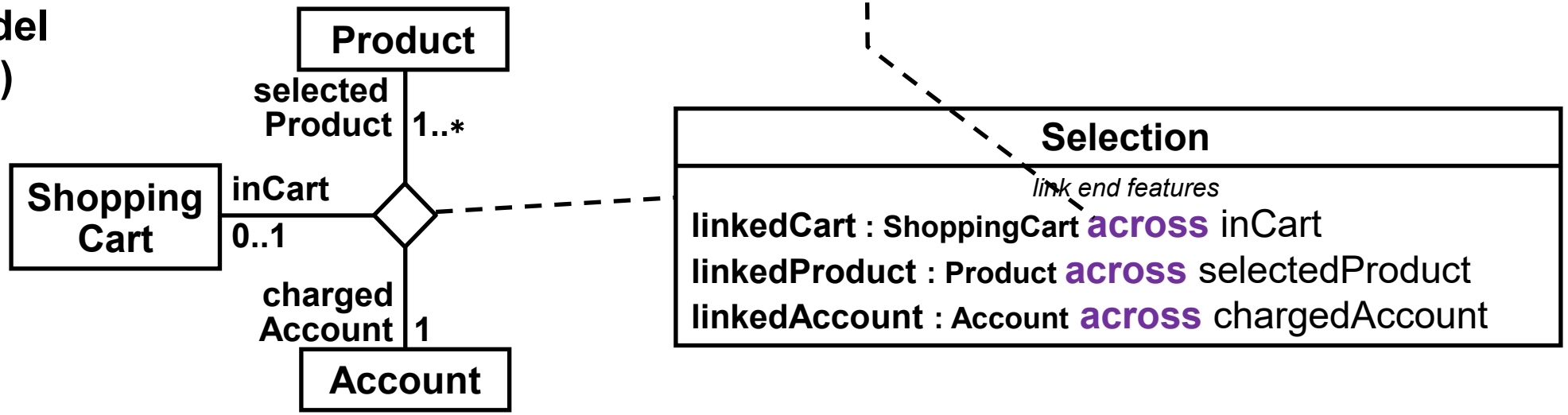
KERML-41

For n-ary associations:

- **source must be link end feature that identifies a cross feature**
- **target must chain through sequence of other "n-1" link end values, then cross feature**

Metamodel (M2)

Model (M1)




```
assoc Selection {
  linkEnd [0..1] linkedCart: ShoppingCart across inCart;
  linkEnd [1..*] linkedProduct: Product across selectedProduct;
  linkEnd [1] linkedAccount: Account across chargedAccount;}
```