Notations

* Eléments syntaxiques pris du méta-modèle
* Δ : environnement contenant les types déclarés
  + Forme indéterminée pour l’instant
  + Construction indéterminée pour l’instant
* Γ : environnement contenant les types des variables
* Φ : environnement contenant les fonctions déclarées
* Σ : environnement contenant les systèmes déclarés
* Μ : environnement contenant les médiateurs déclarés
* φ : type d’une fonction
* σ : type d’un système
* μ : type d’un médiateur
* α : type d’une architecture
* τ : type d’une expression
* τ1 <: τ2 : τ1 sous-type de τ2 (inclus le cas τ1 même type que τ2)
* V : liste de Valuing
* P : listes des types des paramètres formels d'un behavior

Jugements

* Δ,Φⱶd bien typé : dans l’environnement Δ,Φ le type d est bien typé
* Δ,Φⱶf:φ : dans l’environnement Δ,Φ la fonction f a le type φ
* Δ,Φⱶs:σ : dans l’environnement Δ,Φ le système s est de type σ
* Δ,Φⱶm:μ : dans l’environnement Δ,Φ le médiateur m est de type μ
* Δ,Φ,Σ,Μⱶa:α : dans l’environnement Δ,Φ,Σ,Μ l’architecture a est de type α
* Δ,Φⱶe:τ : dans l’environnement Δ,Φ l’expression e est de type τ

D'après le diagramme SoSADL

D'après le diagramme EntityBlock

D'après le diagramme DataType

→ ajouter la condition *vmin*<=*vmax*? Non parce que ce n'est pas décidable.

D'après le diagramme FunctionDecl

→ c'est la règle lorsque *type* ET *expression* sont donnés.

→ où exprime-t-on le fait que le nom déclaré dans Valuing ne doit pas avoir déjà été déclaré dans le même scope ?

→ Valuing avec type non donné et expression donnée.

Note : l'expression est toujours donnée (imposée par la grammaire concrète).

→ Valuing vide (tous les Valuing ont été traités, [] = liste vide)

D'après le diagramme SystemDecl

D'après le diagramme MediatorDecl

D'après le diagramme GateAndDutyDecl

D'après le diagramme ArchitectureDecl

D'après le diagramme ArchBehaviorDecl

D'après le diagramme Behavior

→ le repeat ne peut pas être suivi d'un statement

→ le choose ne peut pas être suivi d'un statement

→ IfThenElse sans else

→ IfThenElse avec else

→ est-ce que name intervient dans le typage ? Quelle est sa sémantique ?

→ est-ce que AskAssertion lie des variables, et si oui lesquelles ?

→pour l'instant E n'est pas défini :

* faut-il typer l' « l'environnement » qui contient les contraintes ? Auquel cas cela pourrait être E
* sinon, il faut introduire une variable de type pour chaque nom. Mais dans la pratique cela sera-t-il vraiment utilisable ?
  + Par exemple « ask toto is (x=y) » ne permettra de conclure rien d'autre que le fait que x et y sont du même type. Cependant les autres instructions pourraient fournir suffisamment de contraintes pour déterminer ce type.
  + Par exemple « ask toto is (x=y) value z = x+1 » : rien ne permettrait de s'assurer que les types correspondent bien au tell correspondant « value x = sequence{1} tell toto is x=y ». Cet exemple contient un problème de types.

"

D'après le diagramme Expression

ComplexName à faire récursivement selon le modèle de Valuing.

→ à faire : le cas où left, ou right, ou left et right, sont RangeType (vmin, vmax). Problème : dans le cas où left ou right est de type IntegerType(), on ne peut pas calculer statiquement les bornes du RangeType résultat.

→ object a un type déclaré, method existe dans l'environnement des fonctions (= méthodes) du type de object, les paramètres (effectifs) sont bien typés, le type de l'expression MethodCall() est le type de cette méthode.

→ soit une expression Tuple{e1, e2..., en}, on devrait pouvoir déterminer précisément le type τe1, τe2..., τen de chaque TupleElement e1, e2... en et en déduire que le type de Tuple{e1, e2..., en} est Tuple(τe1, τe2..., τen).

→ soit une expression Sequence{e1, e2..., en}, on devrait pouvoir déterminer précisément le type τe1, τe2..., τen de chaque élément e1, e2... en et en déduire que le type de Sequence{e1, e2..., en} est Sequence(τe1, τe2..., τen).

→ object est de type Sequence(τ) qui est déclaré, expression est du type τe, le type de l'expression Map(object,expression) est donc de type Sequence(τe ).

→ object a un type déclaré, field est un nom qui existe dans l'environnement des variables et des fonctions du type de l'objet, le type de ce field est τ, le type de l'expression Field(object,field) est donc le type du field.

D'après le diagramme Assertion

→ à faire : le cas où left, ou right, ou left et right, sont RangeType (vmin, vmax). Problème : dans le cas où left ou right est de type IntegerType(), on ne peut pas calculer statiquement les bornes du RangeType résultat.

D'après le diagramme Protocol

→ devrait tenir compte de la séquentialité (ordre) des statements.

→ IfThenElseProtocol sans else

→ IfThenElseProtocol avec else

→ comment indiquer que Action est TellAssertion ou AskAssertion ?

DoExpr() et Done() déjà traité dans les expressions.