Masao Ito (NIL)
Jan/7/2020

SPARK/Ada 英日辞書

SPARK/Ada English-Japanese Dictionary

はじめに

SPARK/Ada における新しい概念は，単に日本語に置き換えただけでは，理解することができません．日本語が示す範囲と異なる可能性があります．場合によって，カタカナでそのまま受け入れる必要があります．

必要と思える箇所には，簡単な説明を付けています．

A/S は，Ada と SPARK の略です．A は Ada言語で用いるもの．S は SPARK で用いるものです．SPARK は，Ada 言語のサブセットなので，Ada 用語も利用します．

参照先は，以下です．

ARM: ISO/IEC 8652:2012(E) Ada Reference Manual

SUG: SPARK 2014 User's Guide https://docs.adacore.com/spark2014-docs/html/ug/index.html

（English Version）

SPARK/Ada

English-Japanese dictionary

Introduction

The new concepts in SPARK/Ada cannot be understood by simply replacing them with Japanese. It may be different from the meaning indicated in Japanese. In some cases, you may need to accept katakana as is.

Where it seems necessary, a brief description is provided.

A/S stands for Ada and SPARK. A is the term used with the Ada language. S is for use with SPARK. SPARK is a subset of the Ada language, so it also uses Ada terminology.

References

ARM: ISO/IEC 8652:2012(E) Ada Reference Manual

SUG: SPARK 2014 User's Guide https://docs.adacore.com/spark2014-docs/html/ug/index.html

A

英語	日本語	A/S	補足
access type	アクセス型	A	Cにおけるポインタと同様，ただ，それ自身が型であることに注意が必要である．ARM 3.10
aggregate	集成式	A	一般の配列およびレコード型（構造体）に対して値を与える方法．array aggregate といえば，配列集成式．ARM 4.3
alias	別名（化）	A	同一オブジェクトを指す複数の名前のこと．エイリアス．明示的に別名を作成する aliased 予約語がある ARM 3.3.1． SPARK では，別名は証明の妨げとなるため，使用しないことを推奨している．
array	配列	A	同一の副型を要素として持つ複合型．添字と要素において型定義が可能である．Cなどの配列とは異なっている．ARM 3.6
assignment	代入	A	オブジェクトの値を，右辺データ表現の評価結果に置き換える．制限付き副型への代入時には，範囲検査が行われる．ARM 5.2
aspect	アスペクト	A	Ada 2012 で導入された概念．with節とともに示す．SPARKは，契約表現等をこのアスペクトを用いて行う．アスペクト指向プログラムのアスペクトとは無関係である
assertion/ assert	表明	A	契約を示す事前・事後条件等．契約には，「表明」が必要である．プラグマassert文は，証明時に検査対象となる SUG 5.6.1
assumption	仮定／前提	S	「スイッチ --assumptions が設定されたとき, GNATprove は，その結果ファイル gnatprove.out 中の残っ ている仮定についての情報を出力します.」SUG 6.3.5

```
type Integer_Access is access all Integer;
V : Integer_Access := new Integer;
```

アクセス型の例

```
type Singleton is record
   V : Integer;
end record;
V1 : Singleton := (V => 0);
```

集成式の例

B

英語	日本語	A/S	補足
body	ボディ部／実体部	A	単にボディ部と呼ばれる場合もある．仕様部に対応し，仕様部記述に対する実現を記述する

C

英語	日本語	A/S	補足
check	検査／検査項目	S	例えば，GNATprove による検査を指す
child package	子パッケージ	A	パッケージ内に包含されるパッケージ．柔軟に入れ子構造を取れるという特徴がある
composite type	複合型	A	レコード型，配列型などの総称（ARM 3.2）JISXでは，構造型という訳語を与えている．Cなどの構造体との関係が連想されるため，避けている
component	コンポーネント／要素	A	一般的には，コンポーネントとカタカナにする．配列要素を指す場合もあり，そのときは単に要素を指す
completion	完結化	A	実行時準備処理時に，必要な情報を補うこと
consequence	帰結	S	特別な使い方としては，ケース文でガード条件が真の場合に選ばれる式を指す場合がある
constrained array	制限付き配列型	A	添字範囲が確定している配列型
constraint	制約	A	副型の宣言時に加える制限のこと． ARM 3.2.2
contract	契約	S	契約型設計に由来する． 副プログラム契約：副プログラム（手続き・関数）がサービスを提供するためには，事前条件を満足していないといけない．その代わりに手続き・関数は，事後条件を満足するということを「契約」として表現する． パッケージ契約：広域変数の状態・初期化，パッケージの初期化を規定する 型契約：スカラー型範囲，レコード型区別子，述語
controlled type	被制御型	A	タグ付き型の一つで，インスタンスを開放するときに（finalization），ユーザ指定による特別な処理を必要とする場合に用いる型．ARM 7.6

D

英語	日本語	A/S	補足
derived type	導出型	A	他の型によって定義された型．ARM 3.4
discrete type	離散型	A	曜日や交通信号の色といった離散的な値をとる型．「型（type）」の項参照．ARM 3.5.5, 3.6.1
discriminant	区別子	A	複合型のパラメータ化に用いる．判別子という訳語を用いる場合もある．オブジェクトの生成時には，制約を加える必要がある．ARM 3.7

```
type Shape_Kind is (Circle, Line, Torus);

type Shape (Kind : Shape_Kind) is record
   X, Y : Float;
   case Kind is
      when Line   =>
         X2, Y2 : Float;
      when Torus  =>
         Outer_Radius, Inner_Radius : Float;
      when Circle    =>
         Radius : Float;
   end case;
end record;
```

区別子の利用例（パラメータ化されたレコード型）

E

英語	日本語	A/S	補足
elaboration	実行時準備処理	A	メインが呼ばれる前に行われる実行系の処理を指す
enclosing construct	内包構成要素	A	変異レコード型における構成要素
entry family	エントリー属	A	タスキングにおける同一名のエントリー集合
enumeration type	列挙型	A	Cと異なり位置に対する整数は与えられない．「型（type）」の項参照．ARM 3.5.1
expression	式	A	ARM 4.4
expression function	式関数	A	仕様部で実体部の内容を内包している小さな関数のこと．ARM 6.8

F

英語	日本語	A/S	補足
failed	不合格となった	S	証明後のメッセージに表示される場合がある．通常は反例とともに示される
flow analysis	フロー分析	S	データフローを検査すること．SUG 5.2.5
formal parameter	仮パラメータ	A	一般の言語における仮引数に相当する
formal verification	形式検証	S	何らかの数学的モデルに基づいて仕様の（表明）内容が実現されていることを証明すること．SPARKにおいては，演繹的方法を用いる
function	関数	A	副プログラムの一つ．返り値を持ち，返り値をムシすることはできない．ARM 6.5

G

英語	日本語	A/S	補足
generic unit	汎用体	A	一種のテンプレートであり，パラメータ化することができる．汎用副プログラムないしは，汎用パッケージを作ることができる．C++等のテンプレートは，この汎用体概念を取り入れたもの．Adaは強い型付けを特徴とするため，汎用体を使わないとさまざまなパッケージを作ることになってしまう．総称体．ARM 12
ghost code	ゴーストコード	S	主として検証を容易にするために用いるコード．スイッチによりコンパイル時には削除できるために，ゴーストという名がある．ゴースト関数，ゴースト変数，ゴースト型，ゴースト手続き等がある．SUG 5.5.10
global	広域	S	スコープが広域であること．フロー分析用のデータ依存に関するアスペクト記述にも用いる

H

英語	日本語	A/S	補足
hold	妥当性を持つ	S	

I

英語	日本語	A/S	補足
illegal	規則違反	-	文法規則に違反していること
index	添字	A	配列における添字として，或いは，タスキングでのエントリー属における添字として使用する ARM 9.1
instance	実体／インスタンス	A	オブジェクト指向で使われるインスタンスと基本的には同じ．オブジェクトの項も併せて参照のこと
instantiate	実体化	A	汎用体を実体化すること．ARM 12.3
integrity	完全性	-	安全性や信頼性を強く要求されるシステムを，High Integrity System と呼ぶ．ARM Annex H
invariant	不変条件	A	実行中に変化しない変数等の値．ループ不変条件，型不変条件がある．ループ不変条件（loop invariant）および型不変条件（type invariant）の項参照

J

英語	日本語	A/S	補足
justify	正当化	S	証明器で証明が困難だが，他の手段によって明らかに正しいと分かっている場合に，注記を付加することで，証明範囲から，該当箇所を外すことが可能である．pragma Annotateを用いることで，代替できる． SUG “Uses of Pragma Annotate GNATprove”

L

英語	日本語	A/S	補足
lemma	補題	S	SPARKには，専用の補題のライブラリがある．SUG 5.10.3．補題とは，（数学的には）既に証明済みの定理であり，証明したいこと（定理）から見ると補助的な役割を果たすものである．ここでは，既に明らかであるものとして，証明器に対してユーザが与える補助的定理のことを指している．それ自身が実際に正しいかどうかを示すのは，ユーザの責務である．
limited type	限定型	A	等値演算子を持たず，同一型同士であっても代入ができない型 ARM 7.5
linked list	連結リスト	S	SPARKライブラリに含まれるデータ構造の一つ
loop invariant	ループ不変条件	S	表明の一つ．ループ中で変化しないプロパティを示す．SUG 5.6.3
loop variant	ループ変化条件	S	表明の一つ．ループ中でのスカラー値の変化（増加・減少）を規定する．SUG 5.6.4

```
Prop := False;
for J in 1 .. 10 loop
   pragma Loop_Invariant (Prop);
   Prop := True;
end loop;
```

ループ不変条件

最初のループでは不変条件はFalseになるが，それ以降ではTrueとなる

```
procedure Terminating_Loops (X : Natural)
with SPARK_Mode
is
   Y : Natural;
begin
   Y := 0;
   while X - Y >= 3 loop
      Y := Y + 3;
      pragma Loop_Variant (Increases => Y);
   end loop;

   Y := 0;
   while X - Y >= 3 loop
      Y := Y + 3;
      pragma Loop_Variant (Decreases => X - Y);
   end loop;
end Terminating_Loops;
```

ループ変化条件

上記の例では，増加・減少の変化条件は全て真となる

M

英語	日本語	A/S	補足
memory corruption	メモリ不斉	-	
modular type	剰余型	A	剰余の値を返す型．「型（type）」の項参照．ARM 3.5.4
modularity	モジュール性	-	

N

英語	日本語	A/S	補足
null	ヌル	A	予約語．ARM 2.9. 例えばアクセス型を初期化しない場合，指している領域は，ヌルとなる　ARM3.3.1
null procedure	ヌル手続き	A	実体部を持たない手続き宣言の簡略版．ARM 6.7
null record	ヌルレコード型	A	要素を持たないレコード型．record null; end record に同じ．ARM 3.8

O

英語	日本語	A/S	補足
object	オブジェクト	A	一般には変数の意味で用いる．あるメモリ領域を占めるものを指す．オブジェクト指向でいうオブジェクト・インスタンスと同じではありません．かつては，「算体」という訳語与えられていた
overload	オーバーロード	A	副プログラムオーバロードや演算子オーバロードといった一般的なオブジェクト指向プログラムにおける意味と同じ．ARM 6.4, 6.6

P

英語	日本語	A/S	補足
package	パッケージ	A	Ada言語におけるプログラム単位．ARM 7
parmeter	パラメータ	A	引数
part	領域／部	A	例えば，body part はボディ部と呼ぶ
postcondition	事後条件	A	手続き・関数が実行後に満足しているべき条件．契約の項参照．ARM 6.1.1
pragma	プラグマ	A	一般的なプラグマと同じ．コンパイラに対する指示を示す．ARM 2.8 pragmas
precondition	事前条件	A	手続き・関数が実行する前に満足すべき条件を示す．ARM 6.1.1
predicate	述語	A	副型のプロパティを表現するために用いる．ARM 3.2.4 subtype predicate
private	非公開	A	かつては，密閉型という訳語が与えられていた．プライベートとカタカナにする訳語もある．ARM 7.3
procedure	手続き	A	副プログラムの一つ．関数と異なり返り値を持たない．ARM 6.5.1
proof	証明	S	演繹的に命題或いは論理式の正しさを示すこと．プログラムにおいては，表明に記された命題或いは論理式が，実装と合っていることを示す
prover	証明器	S	例えば，CVC4などの定理証明ツールを指す．SUG “SPARK architecture, quality assurance and maturity”
protected type	保護型	A	複合型の一つ．その要素は，複数のタスクから並行的にアクセスされる保護操作を通じてのみアクセス可能である
public	公開	A	スコープの一つ．パッケージにおいては，デフォルトのスコープ．パッケージ利用者は，その仕様部で公開されている情報を利用できる

Q

英語	日本語	A/S	補足
qualification	型限定	A	ある型の所属を示す．例えば P.T とするとき，型 T は,パッケージ P で定義している T 型に限定される

R

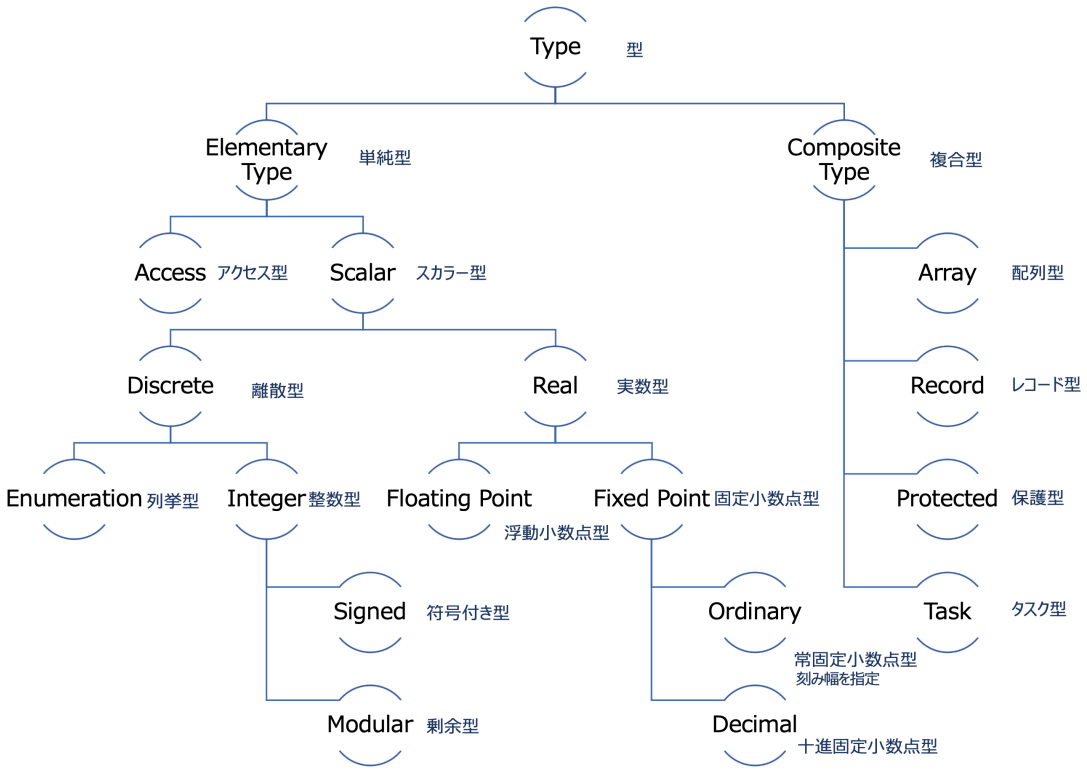
英語	日本語	A/S	補足
record	レコード	A	構造体に類似している．歴史的には「記録」が訳語として用いられた．ARM 3.8
representation	データ表現	A	データ構造表現という意味の場合もある

S

英語	日本語	A/S	補足
scalar type	スカラー型	A	単一の値をとる型．「型（type）」の項参照．ARM 3.5
scope	スコープ	A	可視範囲．一般的な言語と同様の意味を持つが，細かな制御が可能である
short-circuit control	省略制御	A	ブール演算子 and, or は省略制御ではなく，常に両オペランドを評価する．”and then” “or else” は省略制御形式である ARM 4.5.1
signed integer type	符号付き整数型	A	処理系により定まる最小値と最大値を持つ．他の整数型は，剰余整数型．ARM3.5.4
statement	命令文	A	或いは単に「文」
storage pool	記憶域プール	A	オブジェクトの型により，記憶域プールの一部が Allocator により割り当てられる．ARM 13.11
subtype	副型	A	型から導出される新しい型．副型が示す範囲は，導出もとに対して，常に狭い範囲をカバーする
subprogram	副プログラム	A	手続きと関数のこと
suspension	保留	A	suspension object という場合，同期に関係する．Ravenscar特有のオブジェクトを指す（セマフォー）場合がある．

T

英語	日本語	A/S	補足
tagged type	タグ付き型	A	型拡張可能な型のこと．ARM 3.9
type	型	A	Adaにおいては，抽象データ型を指す．型を厳密に定義することで，暗黙的な型変換や勘違いによる異なる型同士の演算を検出する．静的にプログラム誤りをみつけるために型を用いる．このような言語は，「強い型付け（strong-typing」の言語と呼ばれる．基本型については，下図参照
type invariant	型不変条件	S	型の要素が持つ不変条件を示す．例えば，円の中の各点は，中心から一定の距離内にあることを示す．これは個別の座標 X, Y だけでは表現できない．ARM 7.3.2



Ada言語定義基本型（一部：ARM3.2）

```
package Root_Pkg is
  type Root_Type is tagged
    record
      -- component1
      -- component2
      -- etc
    end record;
  procedure Op1(params);
  procedure Op2(params);
  function Op2(params) return Return_Type;
  -- etc
end Root_Pkg.
```

ベース Root_Type 型（タグ付き型）

```
with Root_Pkg; use Root_Pkg;
package Extended_Pkg is
  type Extended_Type is new Root_Type with
    record
      -- component3 (extension)
    end record;
  procedure Op1(params); -- override Op1
  procedure Op4(params); -- new operation
end Extended_Pkg;
```

Root_Type型を拡張した Extended_Type 型

U

英語	日本語	A/S	補足
unconstrained array	無制約配列型	A	範囲指定がない配列
unconstrained type	無制限型	A	範囲指定がない型．例えば，String型

V

英語	日本語	A/S	補足
variant record	変異レコード型	A	要素を可変にできるレコード型. discriminant （区別子）参照