



チュートリアル ATM

< 目的 >

モデル上の欠陥検出（仕様間の矛盾、レンジオーバーフロー）について紹介します。

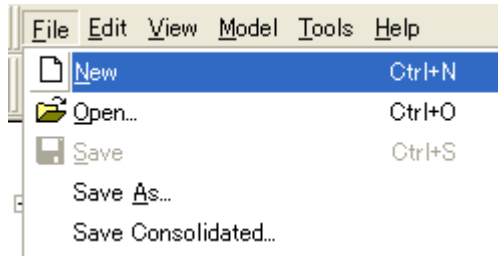
< 要求仕様 >

- ユーザは、預金口座から現金を下ろすことができる。
- ユーザは、口座から他の口座へ送金できる。
- 提携金融機関の場合、2ドルの手数料がユーザに掛かる。

< インタフェース >

- 入力：取引種別、ユーザの利用可能残高、要求された金額、ユーザの機関種別
- 出力：ユーザの新残高

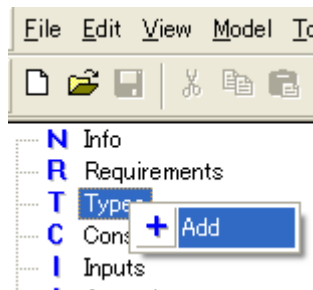
1. TTM を起動した後 新しい TTM モデルを作成します。



2. atm.ttm で保存します。

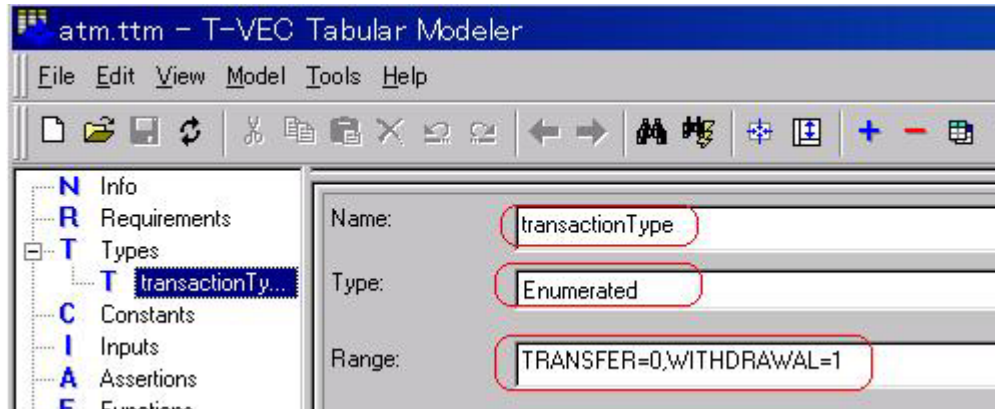
c:~\TVEC_tutorial\exercises\ATM\atm.ttm

3. 型 **transactionType** を作成します。



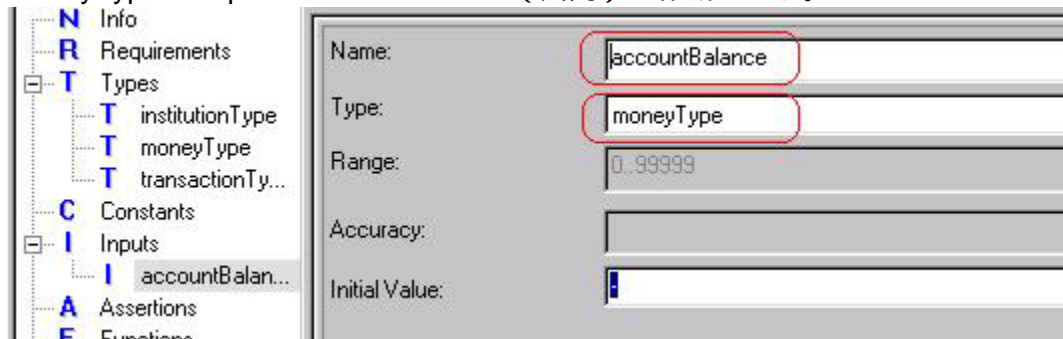
Type : Enumerated

Range : TRANSFER (送金), WITHDRAWAL (引落とし)



4. 同様に 型 **institutionType** を作成します。
 Type : Enumerated
 Range : LOCAL_INSTITUTION, AFFILIATE_INSTITUTION
5. 同様に 型 **moneyType** を作成します。
 Type : Integer
 Range : 0..99999

6. moneyType の Input: **accountBalance** (残高) を作成します。



7. 同様に moneyType の Input: **requestedAmount** (要求額) を作成します。
8. 同様に institutionType の Input: **institution** (機関種別) を作成します。
9. 同様に transactionType の Input: **transaction** (取引) を作成します。

< 振舞い >

10. Terms で、moneyType 型の **serviceCharge** を作成します。(この Term は、手数料を計算するのに使用されます)



11. テーブル 1 行目で、Assignment に 0 を入力します。
12. Condition に、"institution = LOCAL_INSTITUTION" を入力します。
13. テーブルの次の行で、Assignment に 2 (ドル)、Condition に "institution = AFFILIATE_INSTITUTION" を入力します。

Behavior: serviceCharge (of type: moneyType)

#	Assignment	Condition
1	0	institution = LOCAL_INSTITUTION
2	2	institution = AFFILIATE_INSTITUTION

14. 振り込み金額を計算するために、Term で moneyType 型の **transferAmount** を作成します。
15. transferAmount のテーブルを定義します。
16. transaction = TRANSFER なら、transferAmount としてユーザ要求額 (requestedAmount) が送金される。

Behavior: transferAmount (of type: moneyType)

#	Assignment	Condition
1	requestedAmount	transaction = TRANSFER

17. ATM ディスペンサーの金額を計算するために、Term で moneyType 型の **dispensedAmount** を作成します。

18. dispensedAmount のテーブルを定義します。

19. transaction = WITHDRAWAL なら、dispensedAmount としてユーザ要求額 (requestedAmount) が引き落とされる。

Name:	dispensedAmount
Type:	moneyType
Range:	0..99999
Accuracy:	
Initial Value:	
Mode Dependency:	<none>

Main Table Variables Inlined Variables Requirements Comment

Behavior: dispensedAmount (of type: moneyType)

#	Assignment	Condition
1	requestedAmount	transaction = WITHDRAWAL

20. moneyType 型の newAccountBalance と呼ばれる Output を作成します。

21. newAccountBalance のテーブルを定義します。

22. transaction = TRANSFER and transferAmount > 0 なら、

newAccountBalance は、 accountBalance - transferAmount - serviceCharge
(新残高 = 残高 - 振込額 - 手数料)

23. transaction = WITHDRAWAL and dispensedAmount > 0 なら、

newAccountBalance は、 accountBalance - dispensedAmount - serviceCharge
(新残高 = 残高 - 引出し額 - 手数料)

24. transferAmount = 0 and dispensedAmount = 0 なら

newAccountBalance = accountBalance

Name:	newAccountBalance
Type:	moneyType
Range:	0..99999
Accuracy:	
Initial Value:	
Mode Dependency:	<none>

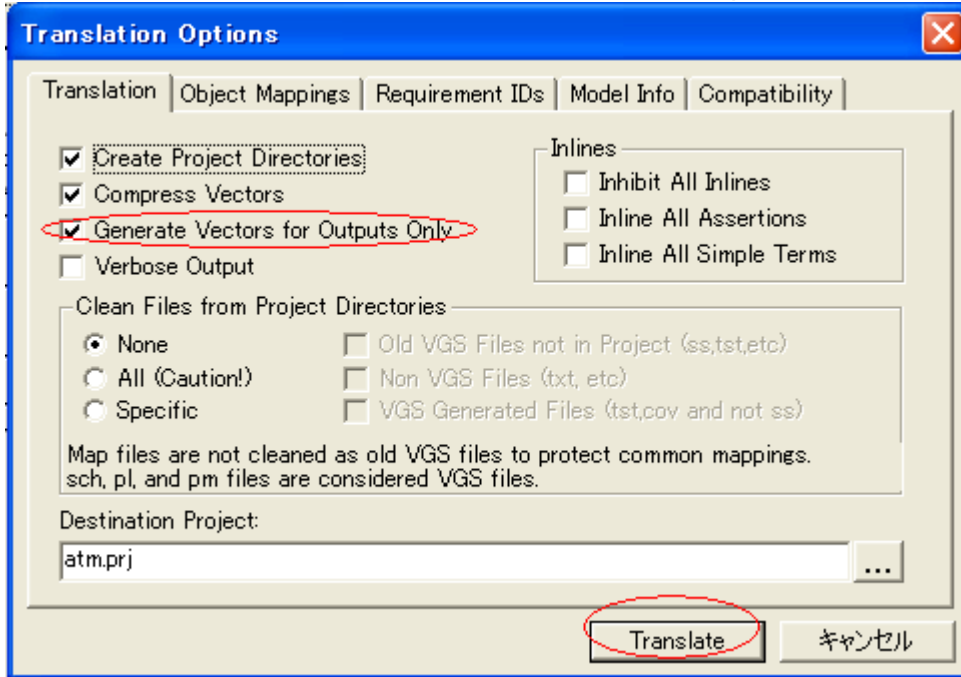
Main Table Variables Inlined Variables Requirements Comment

Behavior: newAccountBalance (of type: moneyType)

#	Assignment	Condition
1	accountBalance - transferAmount - serviceCharge	transaction = TRANSFER AND transferAmount > 0
2	accountBalance - dispensedAmount - serviceCharge	transaction = WITHDRAWAL AND dispensedAmount > 0
3	accountBalance	transferAmount = 0 AND dispensedAmount = 0

<変換とビルド>

25. Tools メニュー -> Translate Model -> Translation Options ダイアログを開きます。



“Generate Vectors for Outputs Only” のチェックボックスを外せば、Term サブシステムのリベクタも生成されます。

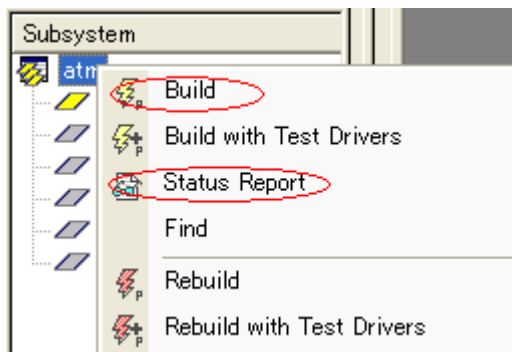
26. Translate ボタンをクリックします。

27. モデルの構文チェックが自動的に行われます。

28. Tools メニュー -> Open in T-VEC を選択し、T-VEC VGS を開きます。

29. プロジェクトをビルドするために、プロジェクト名で右クリックし、メニューから Build を選択します。

30. プロジェクト名で右クリックし、Status Report を選択します。



<仕様間の矛盾・欠陥を検出>

atm Project Status

Project Filename: C:\mbd_course\exercises_fuji\ATM\atm_status.html

Total Test Case Comparisons	0
Total Test Comparison Failures	0
Total Test Vectors	4
Total Test Paths (DCPs)	9
Date Generated	12-30-06 15:07:50

Subsystem	Compilation		Test Vectors		Coverage		Test Results	
	DCPs	Warn/Err	Vectors	Warn/Err	Untested DCPs	Warn/Err	Failures	Comparisons
newAccountBalance	3	0/0	4	2/2	1 of 3	0/7	-	-
_coreRTS	0	0/0	-	-	-	-	-	-
_runTimeData	0	0/0	-	-	-	-	-	-

31. 上記”Untested DCPs”（テストベクタを生成出来ないパスが存在するの意味）の箇所”1 of 3”をクリックすると、レポートが表示されます。

newAccountBalance Coverage Analysis **FAILED**

Vector File: C:\mbd_course\exercises_fuji\ATM\test_vectors\newAccountBalance.TST

Time Run	05-30-07 13:02:28
Analyzer Version	3.4.0
Total Number of DCPs	3
DCPs Not Covered	1
Predicates Requiring Coverage	19
Predicates Not Covered	6 newAccountBalance_FR_3 cv_newAccountBalance_RP_3 newAccountBalance<<3>> newAccountBalance_3_LS newAccountBalance_RP_3 RP3
Total Coverage Errors	7
Total Coverage Warnings	0
Test Generation Failures	2

Uncovered DCP Paths

DCP Number	DCP Path	Failure Detection
3	newAccountBalance<<3>>, newAccountBalance_FR_3, cv_newAccountBalance_RP_3, newAccountBalance_RP_3, newAccountBalance_RP_0, RP3, newAccountBalance_3_LS (goto model)	Vector Generator

32. ここで”goto model”をクリックすると、TTMモデル上の該当箇所が表示されます。

newAccountBalance

Name: newAccountBalance
 Type: moneyType
 Initial Value: -
 Range: 0.99999
 Accuracy:

Table Type
 Condition
 Event
 Inline

Main Mode Dependency Requirements Inlined Variables Comment

Behavior: newAccountBalance

#	Assignment	Condition	Requirement ID	Comment
1	accountBalance - transferAmount - serviceCharge	transaction = TRANSFER and transferAmount > 0		
2	accountBalance - dispenseAmount - serviceCharge	transaction = WITHDRAWAL and dispenseAmount > 0		
3	accountBalance	transferAmount = 0 AND dispenseAmount = 0		

33. newAccountBalance の 3 列目に問題があるようです :
 “ **dispensedAmount = 0 AND transferAmount = 0** “

34. ここでは引き落とし額 (**dispensedAmount**) と送金額 (**transferAmount**) が同時に 0 である場合に新残高に変化が無いことを定義していますが、モデル上に引落としと送金が同時に存在しないことをモデル検査で検出 (証明) しています。テストベクタ生成によるモデル検査であれば、このように要件間の矛盾を検出し、カバレッジエラーとしてレポートすることができます。

Behavior: transferAmount

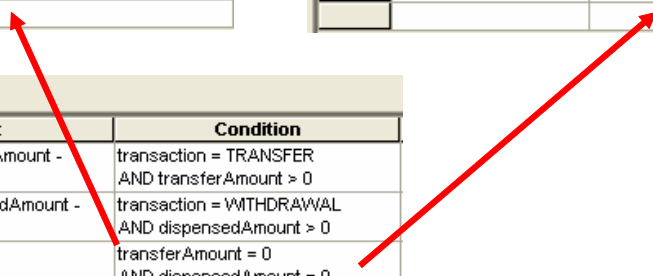
#	Assignment	Condition
1	requestedAmount	transaction = TRANSFER

Behavior: dispensedAmount

#	Assignment	Condition
1	requestedAmount	transaction = WITHDRAWAL

Behavior: newAccountBalance

#	Assignment	Condition
1	accountBalance - transferAmount - serviceCharge	transaction = TRANSFER AND transferAmount > 0
2	accountBalance - dispensedAmount - serviceCharge	transaction = WITHDRAWAL AND dispensedAmount > 0
3	accountBalance	transferAmount = 0 AND dispensedAmount = 0



35. dispensedAmount と transferAmount の Term テーブルに仕様を追加してモデルを訂正してください。

(例：dispensedAmount

transaction = TRANSFER の時、dispensedAmount は 0 を割り当てます)

Terms - transferAmount

Behavior: transferAmount (of type: moneyType)

#	Assignment	Condition
1	requestedAmount	transaction = TRANSFER AND fundsAvailable
2	0	transaction = WITHDRAWAL

Terms - dispensedAmount

Behavior: dispensedAmount (of type: moneyType)

#	Assignment	Condition
1	requestedAmount	transaction = WITHDRAWAL AND fundsAvailable
2	0	transaction = TRANSFER

36. モデルを再変換 (translate) します。

37. テストベクタを再生成します。(Rebuild)

38. Status Report を再生成して確認します。

atm_tutorial_status.html

atm_tutorial Project Status

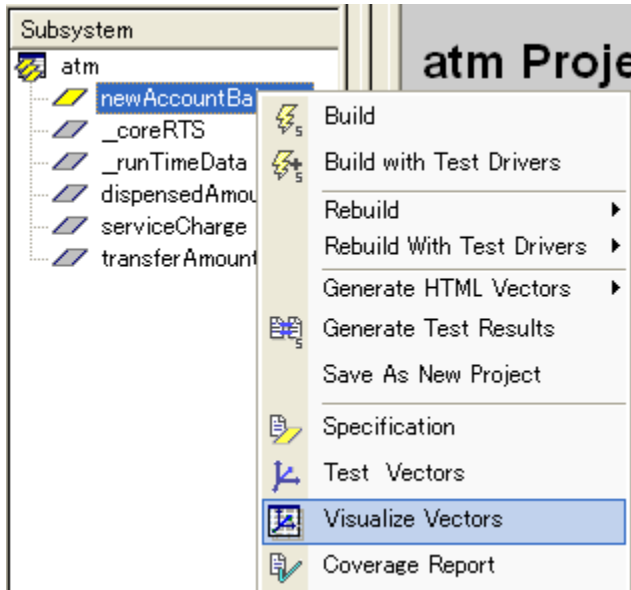
Project Filename: Your filename appears here.

Total Test Case Comparisons	0
Total Test Comparison Failures	0
Total Test Vectors	16
Total Test Paths (DCPs)	9
Date Generated	12-06-06 19:42:04

Subsystem	Compilation		Test Vectors		Coverage		Test Results	
	DCPs	Warn/Err	Vectors	Warn/Err	Untested DCPs	Warn/Err	Failures	Comparisons
dispensedAmount	2	0/0	4	0/0	0	0/0	-	-
newAccountBalance	3	0/0	6	2/0	0	0/0	-	-
serviceCharge	2	0/0	2	0/0	0	0/0	-	-
transferAmount	2	0/0	4	0/0	0	0/0	-	-
__coreRTS	0	0/0	-	-	-	-	-	-
__runTimeData	0	0/0	-	-	-	-	-	-

<レンジオーバーフローの検出>

Status Report において、2つのテストベクタでレンジワーニングが表示されます。
newAccountBalance で右クリックし、Visualize Vectors をクリックすることで、このサブシステムのベクタを表示します。



#	newAccountBalance__OUT	accountBalance	dispensedAmount__VAR	institution	requestedAmount	service
1	9.9998e+004	9.9999e+004	9.9999e+004	(LOCAL_INSTITUTION = 0)	1.0e+000	(
2	-9.9999e+004 OUTSIDE RANGE {0.0e+000..9.9999e+004}	0.0e+000	0.0e+000	(LOCAL_INSTITUTION = 0)	9.9999e+004	(
3	9.9998e+004	9.9999e+004	1.0e+000	(LOCAL_INSTITUTION = 0)	1.0e+000	(
4	-9.9999e+004 OUTSIDE RANGE {0.0e+000..9.9999e+004}	0.0e+000	9.9999e+004	(LOCAL_INSTITUTION = 0)	9.9999e+004	(
5	9.9999e+004	9.9999e+004	0.0e+000	AFFILIATE_INSTITUTION = 1	(0.0e+000)	9.9

2つのテストベクタの出力が、レンジ範囲を超えています(OUTSIDE RANGE)。

その理由は、

- NewAccountBalance は、0~99999 のレンジを持った moneyType です。
- 預金が残額の照会なしで引き出された為に、出力が負になるケースを検出しています。
- 解決策
 39. **fundAvailable** という名前で Boolean 型の新しい Term を作成します。これは、ユーザの残高が、要求額以上かどうか確認するものです。(預金額チェック)
 40. $accountBalance \geq (serviceCharge + requestedAmount)$ なら $fundAvailable = TRUE$

41. $\text{accountBalance} < (\text{serviceCharge} + \text{requestedAmount})$ なら、
 $\text{fundsAvailable} = \text{FALSE}$

Behavior: fundsAvailable (of type: Boolean)

#	Assignment	Condition
1	TRUE	$\text{accountBalance} \geq (\text{serviceCharge} + \text{requestedAmount})$
2	FALSE	$\text{accountBalance} < (\text{serviceCharge} + \text{requestedAmount})$

42. fundsAvailable をチェックするために、transferAmount を更新します。

43. 更新後の transferAmount は、下記ようになります。
 $\text{transaction} = \text{TRANSFER}$ and fundsAvailable なら、
transferAmount は requestedAmount
 $\text{transaction} = \text{TRANSFER}$ and NOT(fundsAvailable) なら、
transferAmount は 0
 $\text{transaction} \neq \text{TRANSFER}$ なら、 transferAmount は 0

Behavior: transferAmount (of type: moneyType)

#	Assignment	Condition
1	requestedAmount	$\text{transaction} = \text{TRANSFER}$ AND fundsAvailable
2	0	$\text{transaction} = \text{WITHDRAWAL}$ AND NOT (fundsAvailable)
3	0	$\text{transaction} \neq \text{TRANSFER}$

44. 同様に、fundsAvailable を参照するよう dispensedAmount も更新します。

Behavior: dispensedAmount (of type: moneyType)

#	Assignment	Condition
1	requestedAmount	$\text{transaction} = \text{WITHDRAWAL}$ AND fundsAvailable
2	0	$\text{transaction} = \text{TRANSFER}$ AND NOT (fundsAvailable)
3	0	$\text{transaction} \neq \text{WITHDRAWAL}$

45. モデルを保存して再変換します。
46. T-VEC プロジェクトファイルをリロードします。
47. テストベクタを再生成します。
48. Status Report を再生成し表示します。（エラーはありません）

atm Project Status
Project Filename: C:\TVEC_Tutorial\exercises\ATM\atm.prj

Total Test Case Comparisons	0
Total Test Comparison Failures	0
Total Test Vectors	6
Total Test Paths (DCPs)	13
Date Generated	08-13-10 10:20:33

Subsystem	Compilation		Test Vectors		Coverage		
	DCPs	Warn/Err	Vectors	Warn/Err	Untested DCPs	Warn/Err	Failures
newAccountBalance	3	0/0	6	0/0	0	0/0	-
dispensedAmount	3	0/0	-	-	-	-	-

49. ベクタを参照します。範囲外エラーが無くなったことを確認ください。

Test Vectors For Subsystem : newAccountBalance
Vector File: C:\TVEC_Tutorial\exercises\ATM\test_vectors\newAccountBalance.TST
[Open Model Report](#)
[Legend For Vector Table](#)

Jump to Page: <Previous Page 1 of 1 Next Page

Test #	Vector #'s	newAccountBalance__OUT	accountBalance	dispensedAmount__VAR	fundsAvailable__VAR	institution
1	1	0	(1)	0	(TRUE = 1)	(LOCAL_INSTITUTION = 0)
2	2	0	99999	0	(TRUE = 1)	(LOCAL_INSTITUTION = 0)
3	3	0	(1)	1	(TRUE = 1)	(LOCAL_INSTITUTION = 0)
4	4	0	99999	99999	(TRUE = 1)	(LOCAL_INSTITUTION = 0)
5	5	0	0	0	(TRUE = 1)	(LOCAL_INSTITUTION = 0)
6	6	99999	99999	0	(TRUE = 1)	(LOCAL_INSTITUTION = 0)

以上

ツールのデモンストレーションを行っています。
ご興味いただける場合は、お手数ですが下記までご依頼頂けると幸いです。



富士設備工業株式会社 電子機器事業部
〒591-8025 大阪府堺市北区長曾根町1928-1
Tel: 072-252-2128 www.fuji-setsu.co.jp