There are two ATM components: ATM-1 and ATM-2.

The **ATM-1** component supports the following operations:

```
create()                    // ATM is created
card (int x, string y)      // ATM card is inserted where x is a balance and y is a pin #
pin (string x)              // provides pin #
deposit (int d);            // deposit amount d
withdraw (int w);           // withdraw amount w
balance ();                 // display the current balance
lock(string x)              // lock the ATM, where x is a pin #
unlock(string x)            // unlock the ATM, where x is pin #
exit()                      // exit from the ATM
```

The **ATM-2** component supports the following operations:

```
create()                    // ATM is created
CARD (float x, int y)       // ATM card is inserted where x is a balance and y is a pin #
PIN (int x)                 // provides pin #
DEPOSIT (float d);          // deposit amount d
WITHDRAW (float w);         // withdraw amount w
BALANCE ();                 // display the current balance
EXIT()                      // exit from the ATM
```

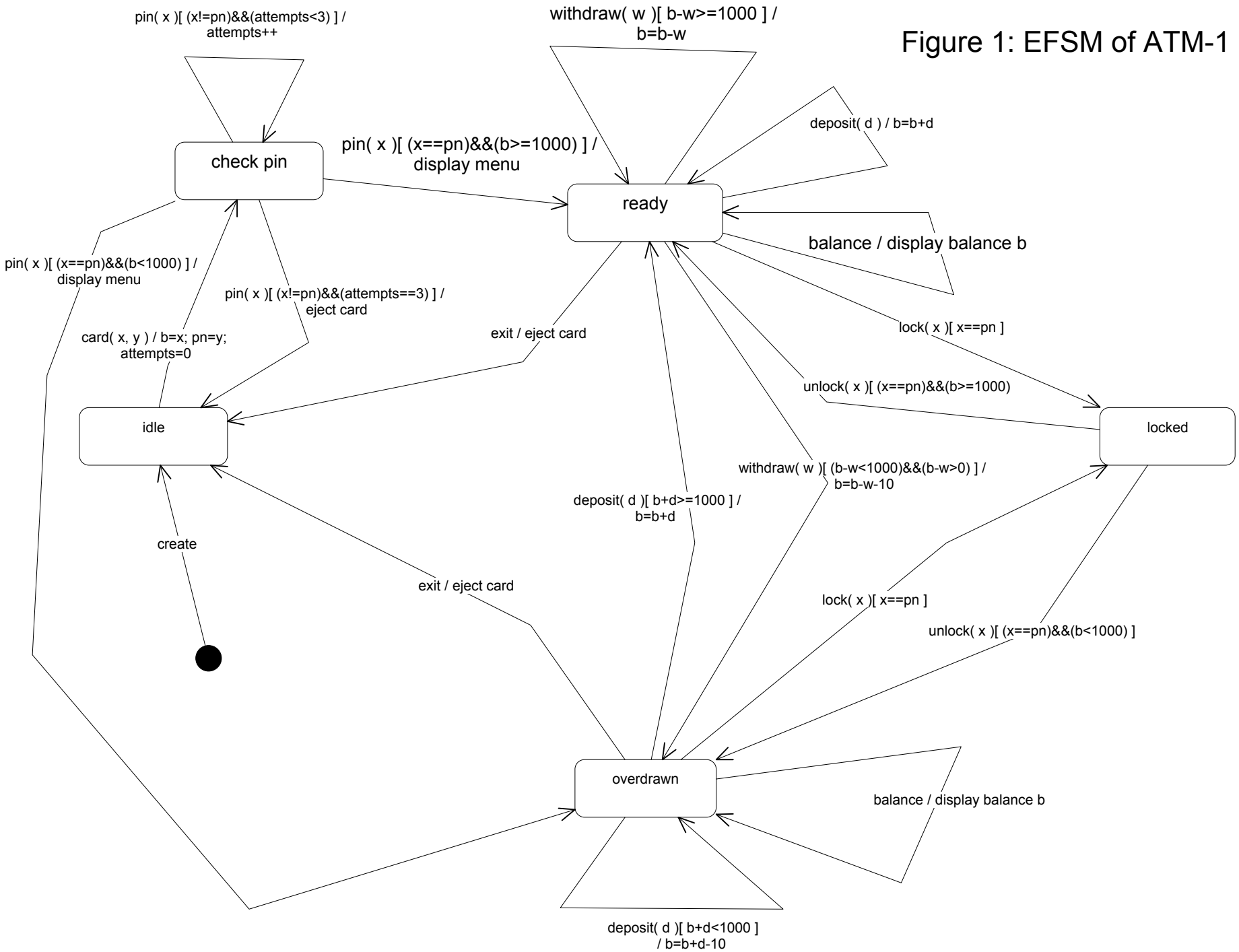
These ATM components are state-based components and support three types of transactions: withdrawal, deposit, and balance inquiry. Before any transaction can be performed, operation *card*(*x*, *y*) (or *CARD*(*x*, *y*)) must be issued, where *x* is an initial balance in the account and *y* is a pin used to get permission to perform transactions. Before any transaction can be performed, operation *pin(x)* (or *PIN(x)*) must be issued. The *pin(x)* (or *PIN(x)*) operation must contain the valid pin # that must be the same as the pin # provided in *card*(*x*, *y*) (or *CARD*(*x*, *y*)) operation. There is a limit on the number of attempts with an invalid pin. The account can be overdrawn (below minimum balance), but a penalty may apply. If the balance is below the minimum balance then the withdrawal transaction cannot be performed. In addition, ATM-1 component can be locked by issuing *lock(x)* operation, where *x* is a pin #. The ATM-1 can be unlocked by *unlock(x)* operation. The detailed behavior of ATM components is specified using EFSM. The EFSM of Figure 1 shows the detail behavior of ATM-1, and the EFSM of Figure 2 shows the detailed behavior of ATM-2. Notice that there are several differences between ATM components.

Aspects that vary between these ATM components:

a. Maximum number of times incorrect pin can be entered
b. Minimum balance
c. Display menu(s)
d. Messages, e.g., error messages, etc.
e. Penalties
f. Operation names and signatures
g. Data types
h. etc.

The goal is to design an executable meta-model, referred to as **MDA-EFSM**, for all ATM components. The MDA-EFSM should capture the "generic behavior" of these two ATM components and should be de-coupled from data and implementation details. Notice that there should be **ONLY** one MDA-EFSM for these two ATM components.

Figure 1: EFSM of ATM-1



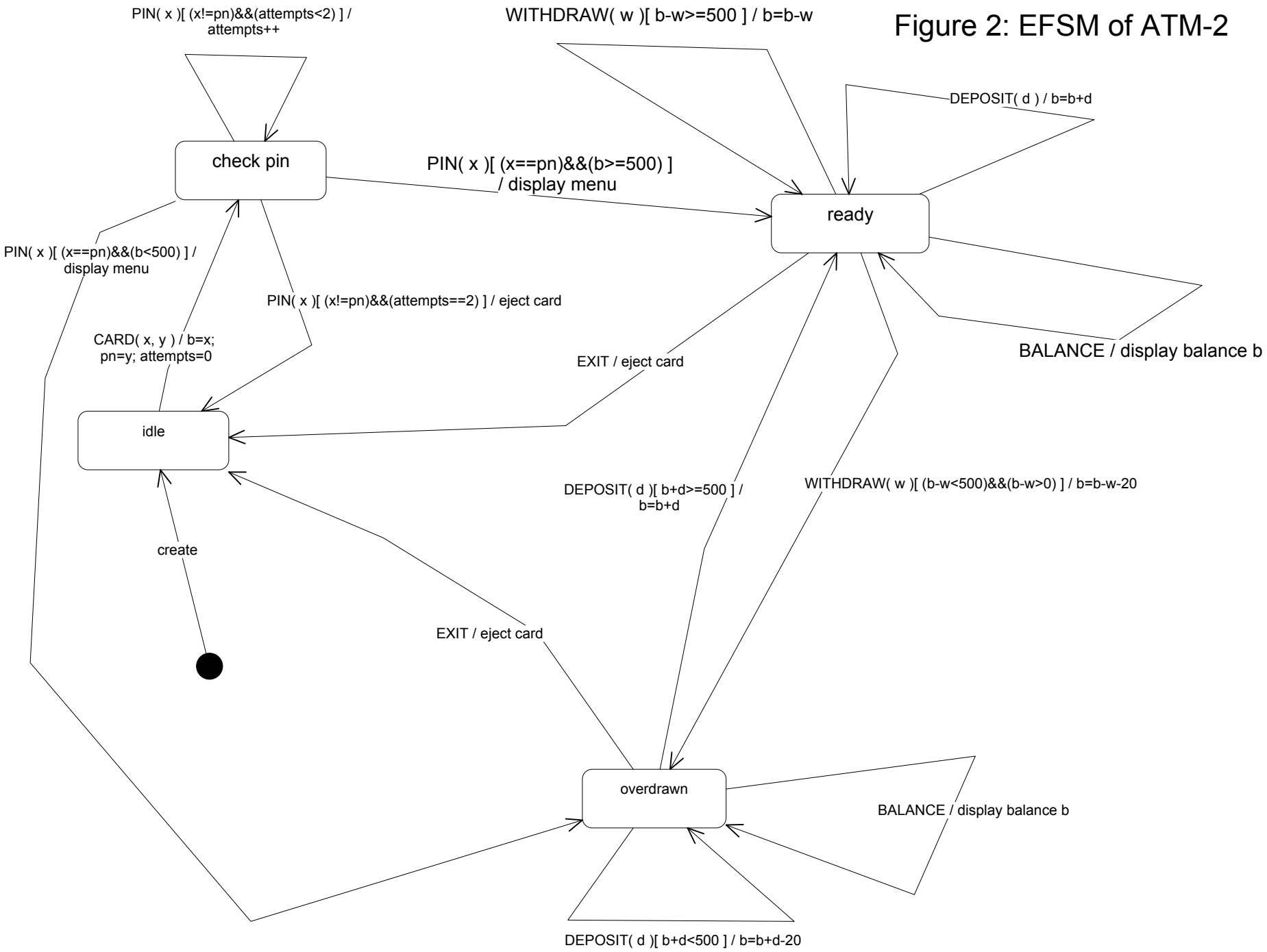
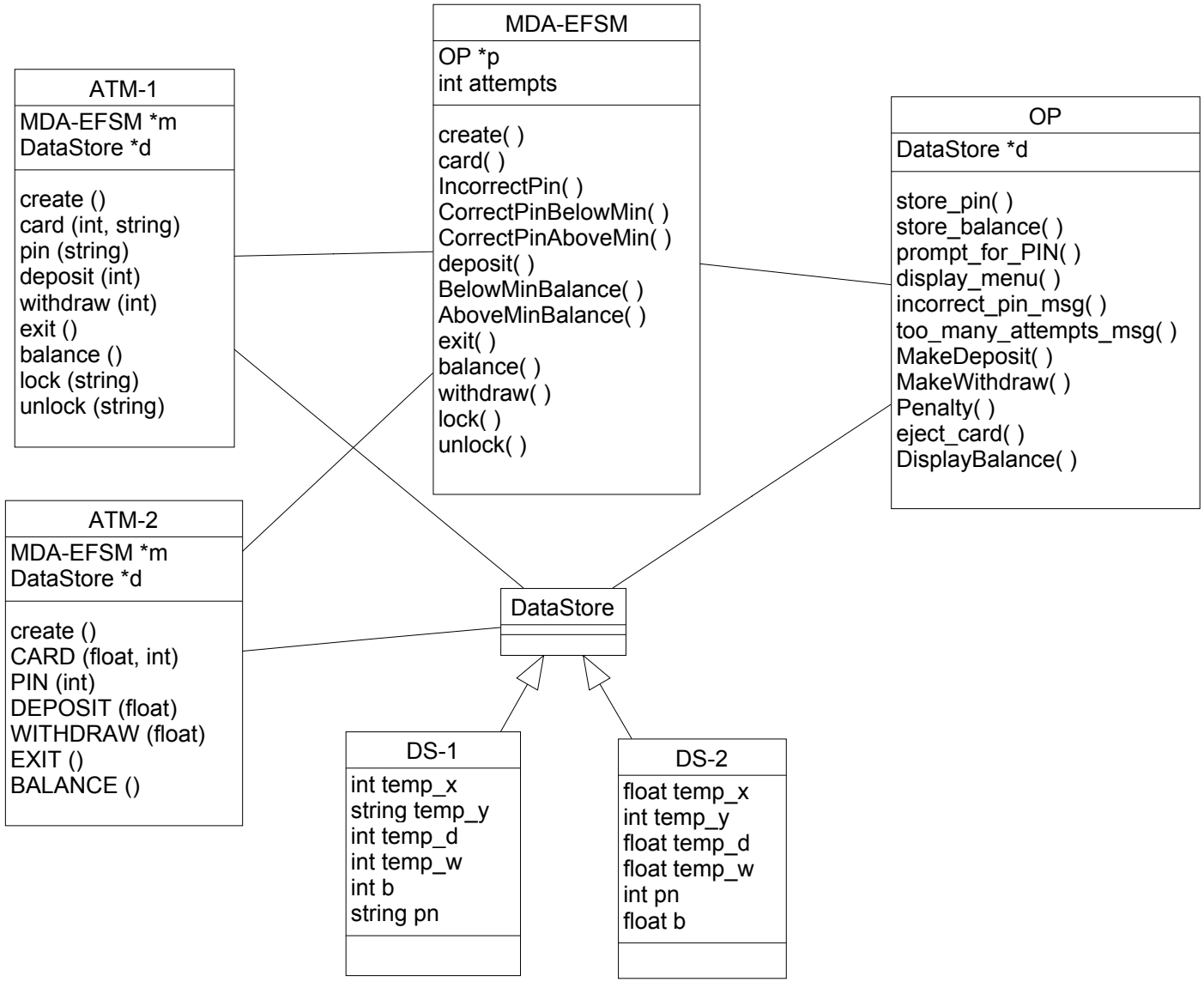
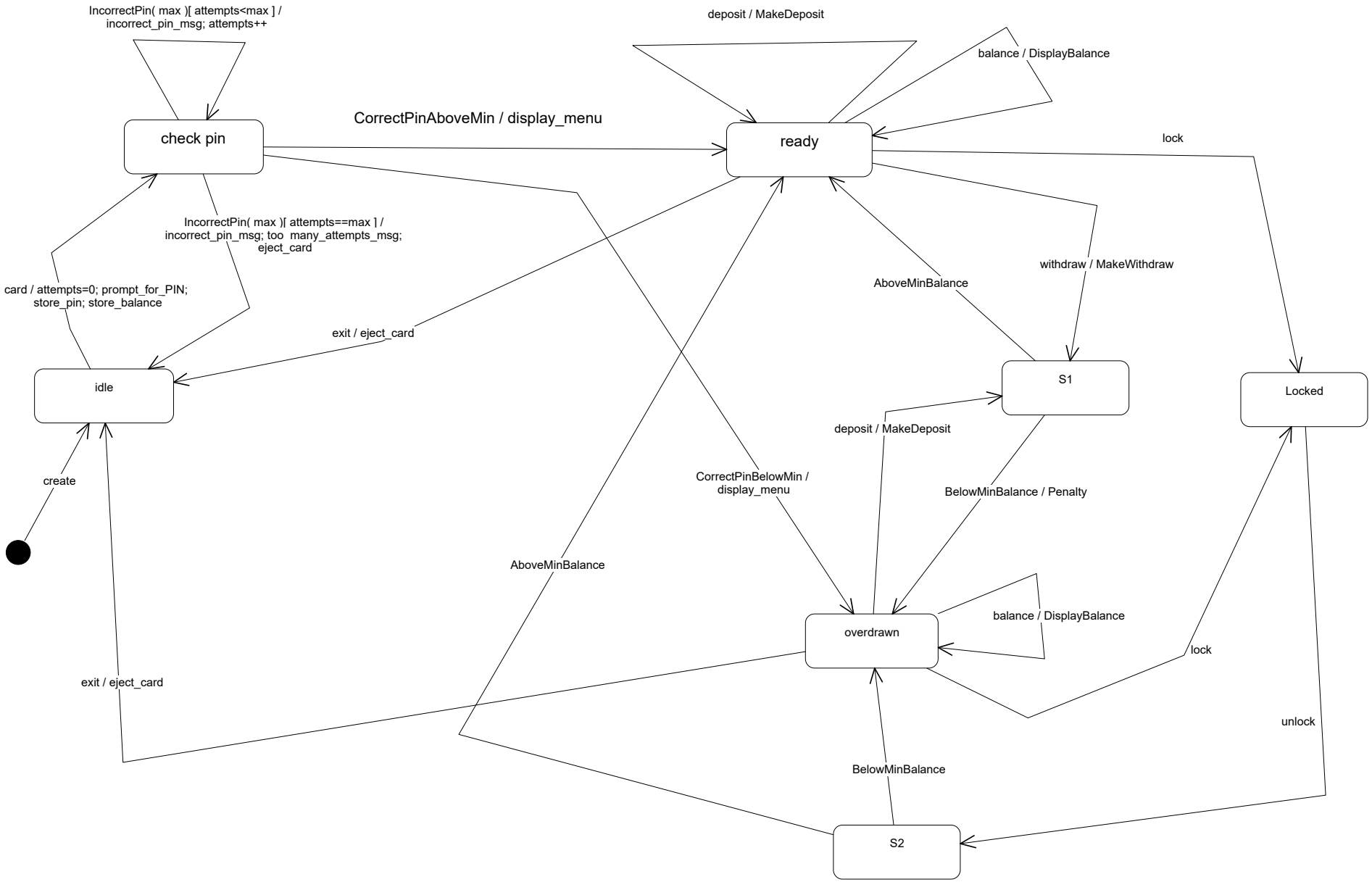


Figure 2: EFSM of ATM-2


MDA-EFSM
OP *p
int attempts
create( )
card( )
IncorrectPin( )
CorrectPinBelowMin( )
CorrectPinAboveMin( )
deposit( )
BelowMinBalance( )
AboveMinBalance( )
exit( )
balance( )
withdraw( )
lock( )
unlock( )
ATM-1
MDA-EFSM *m
DataStore *d
create ()
card (int, string)
pin (string)
deposit (int)
withdraw (int)
exit ()
balance ()
lock (string)
unlock (string)
OP
DataStore *d
store_pin( )
store_balance( )
prompt_for_PIN( )
display_menu( )
incorrect_pin_msg( )
too_many_attempts_msg( )
MakeDeposit( )
MakeWithdraw( )
Penalty( )
eject_card( )
DisplayBalance( )
ATM-2
MDA-EFSM *m
DataStore *d
create ()
CARD (float, int)
PIN (int)
DEPOSIT (float)
WITHDRAW (float)
EXIT ()
BALANCE ()
DataStore
DS-1
int temp_x
string temp_y
int temp_d
int temp_w
int b
string pn
DS-2
float temp_x
int temp_y
float temp_d
float temp_w
int pn
float b


IncorrectPin(max)[attempts<max] /
incorrect_pin_msg; attempts++
deposit / MakeDeposit
balance / DisplayBalance
CorrectPinAboveMin / display_menu
check pin
ready
lock
IncorrectPin(max)[attempts==max] /
incorrect_pin_msg; too many_attempts_msg;
eject_card
withdraw / MakeWithdraw
AboveMinBalance
card / attempts=0; prompt_for_PIN;
store_pin; store_balance
exit / eject_card
idle
S1
Locked
deposit / MakeDeposit
create
CorrectPinBelowMin /
display_menu
BelowMinBalance / Penalty
AboveMinBalance
balance / DisplayBalance
overdrawn
lock
exit / eject_card
unlock
BelowMinBalance
S2

MDA-EFSM Events:
create()
card()
IncorectPin(int max)
CorrectPinBelowMin()
CorrectPinAboveMin()
deposit()
BelowMinBalance()
AboveMinBalance()
exit()
balance()
withdraw()
lock()
unlock

MDA-EFSM Actions:

store_pin	// stores pin from temporary data store to *pin* in data store
store_balance	// stores pin from temporary data store to *b* in data store
prompt_for_PIN	// prompts to enter pin
display_menu	// display a menu with a list of transactions
incorrect_pin_msg	// displays incorrect pin message
too_many_attempts_msg	// display too many attempts message
MakeDeposit	// makes deposit (increases balance by a value stored in temp. data store)
MakeWithdraw	// makes withdraw (decreases balance by a value stored in temp. data store)
Penalty	// applies penalty (decreases balance by the amount of penalty)
eject_card	// ejects the card
DisplayBalance	// displays the current value of the balance

Operations of the Input Processor
(ATM-1)

```
create() {m->create();}

card (int x, string y) {
        d->temp_x=x;
        d->temp_y=y;
        m->card();
}

deposit (int d) {
        d->temp_d=d;
        m->deposit();
        if (d->b < 1000)
              m->BelowMinBalance();
        else m->AboveMinBalance();
}

withdraw (int w) {
        d->temp_w=w;
        if ((d->b-w) > 0) m->withdraw();
        if (d->b<1000)
              m->BelowMinBalance();
        else m->AboveMinBalance();
}

pin (string x) {
        if (x==d->pn) {
              if (d->b<1000)
                    m->CorrectPinBelowMin ();
              else m->CorrectPinAboveMin();
        }
        else m->IncorrectPin(3)
}
```

```
exit() {m->exit();}

balance() {m->balance();}

lock (string x) {
        if (d->pn==x) m->lock();
}

unlock (string x) {
        if (x==d->pn) {
              m->unlock();
              if (d->b<1000)
                    m->BelowMinBalance ();
              else m->AboveMinBalance();
        }
}
```

Notice:
m: pointer to the MDA-EFSM
d: pointer to the data store
In the data store:
b: contains the current balance
pn: contains the correct pin #

Operations of the Input Processor (ATM-2)

```
create() {m->create();}

CARD (float x, int y) {
        d->temp_x=x;
        d->temp_y=y;
        m->card();
}

DEPOSIT (float d) {
        d->temp_d=d;
        m->deposit();
        if (d->b<500)
              m->BelowMinBalance();
        else m->AboveMinBalance();
}

WITHDRAW (float w) {
        d->temp_w=w;
        if ((d->b-w) > 0) m->withdraw();
        if (d->b<500)
              m->BelowMinBalance();
        else m->AboveMinBalance();
}

PIN (int x) {
        if (x==d->pn) {
              if (d->b<500)
                    m->CorrectPinBelowMin ();
              else m->CorrectPinAboveMin();
        }
        else m->IncorrectPin(2)
}

EXIT() {m->exit();}

BALANCE() {m->balance();}
```

Notice:
m: pointer to the MDA-EFSM
d: pointer to the data store
In the data store:
b: contains the current balance
pn: contains the correct pin #