Extensible Canonical Process Model Synthesis Applying Formal Interpretation

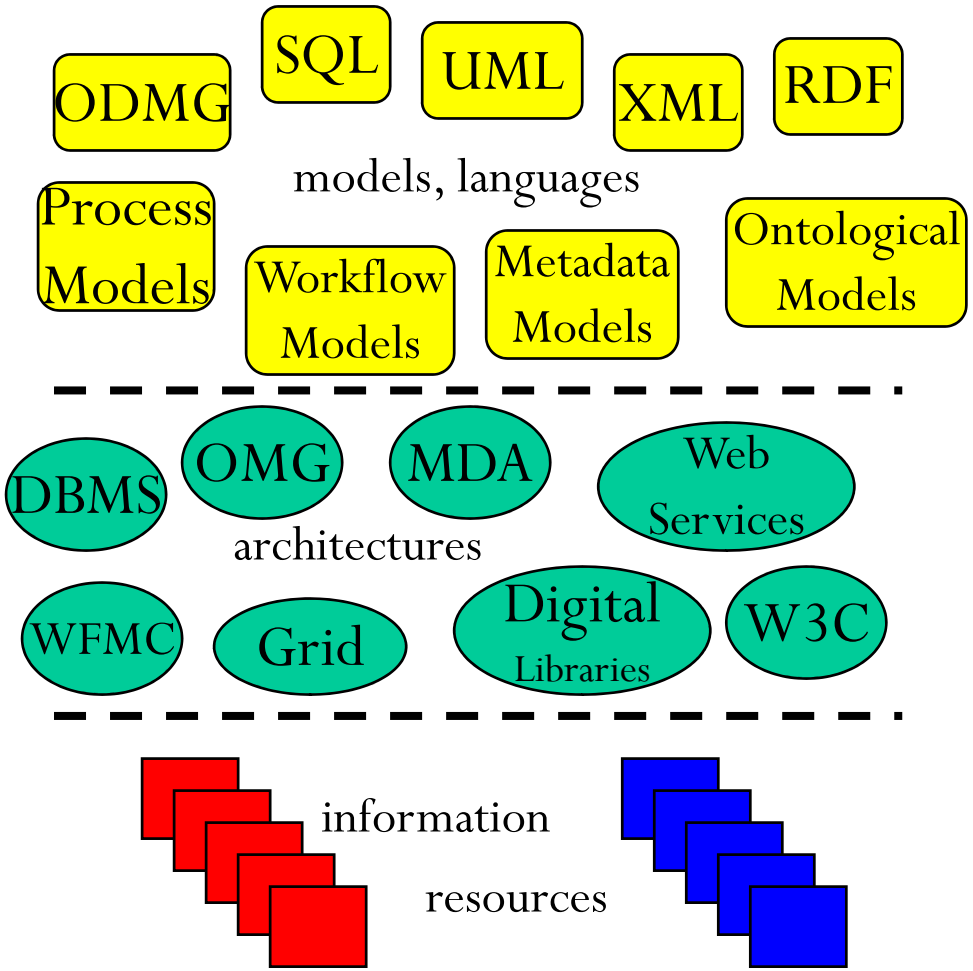
Leonid Kalinichenko, Sergey Stupnikov, Nikolay Zemtsov

Institute for Problems of Informatics
Russian Academy of Sciences

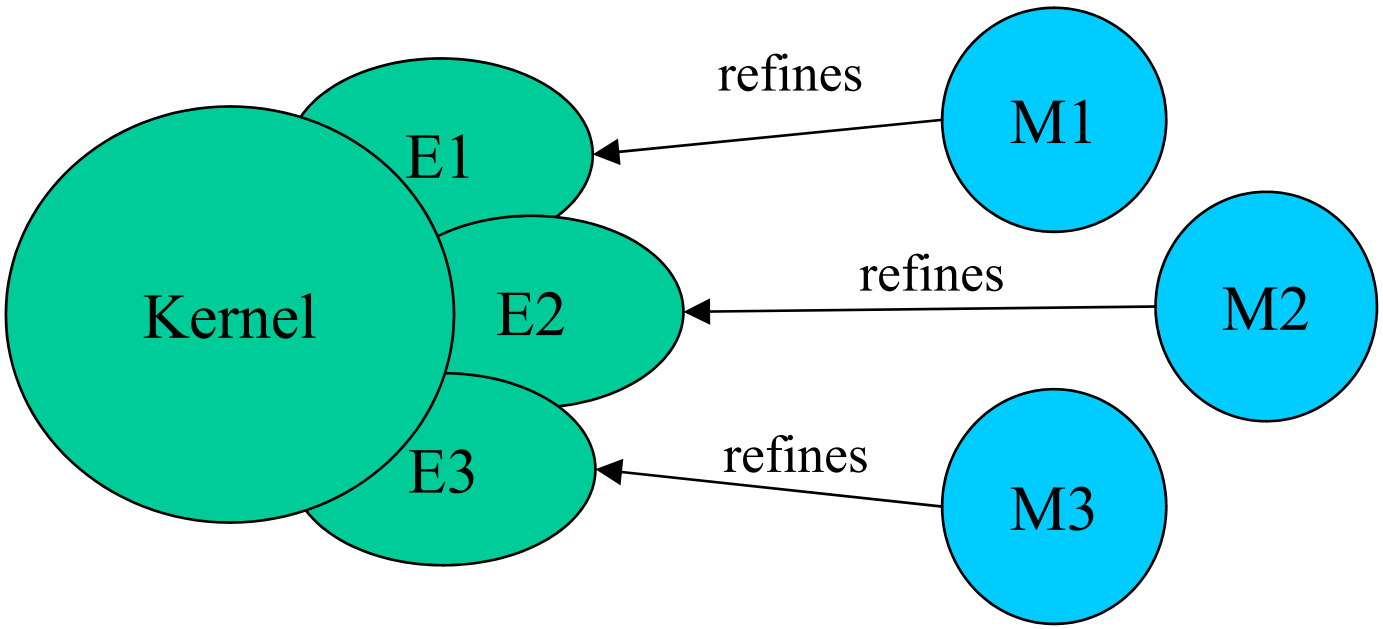
E-mail: {leonidk, ssa, nazem}@ipi.ac.ru

Motivation for the Creation of the Canonical Information Models

- **diversity of information models**
- **need for integration, reuse and composition of information resources**
- **accumulation of heterogeneous information resources**

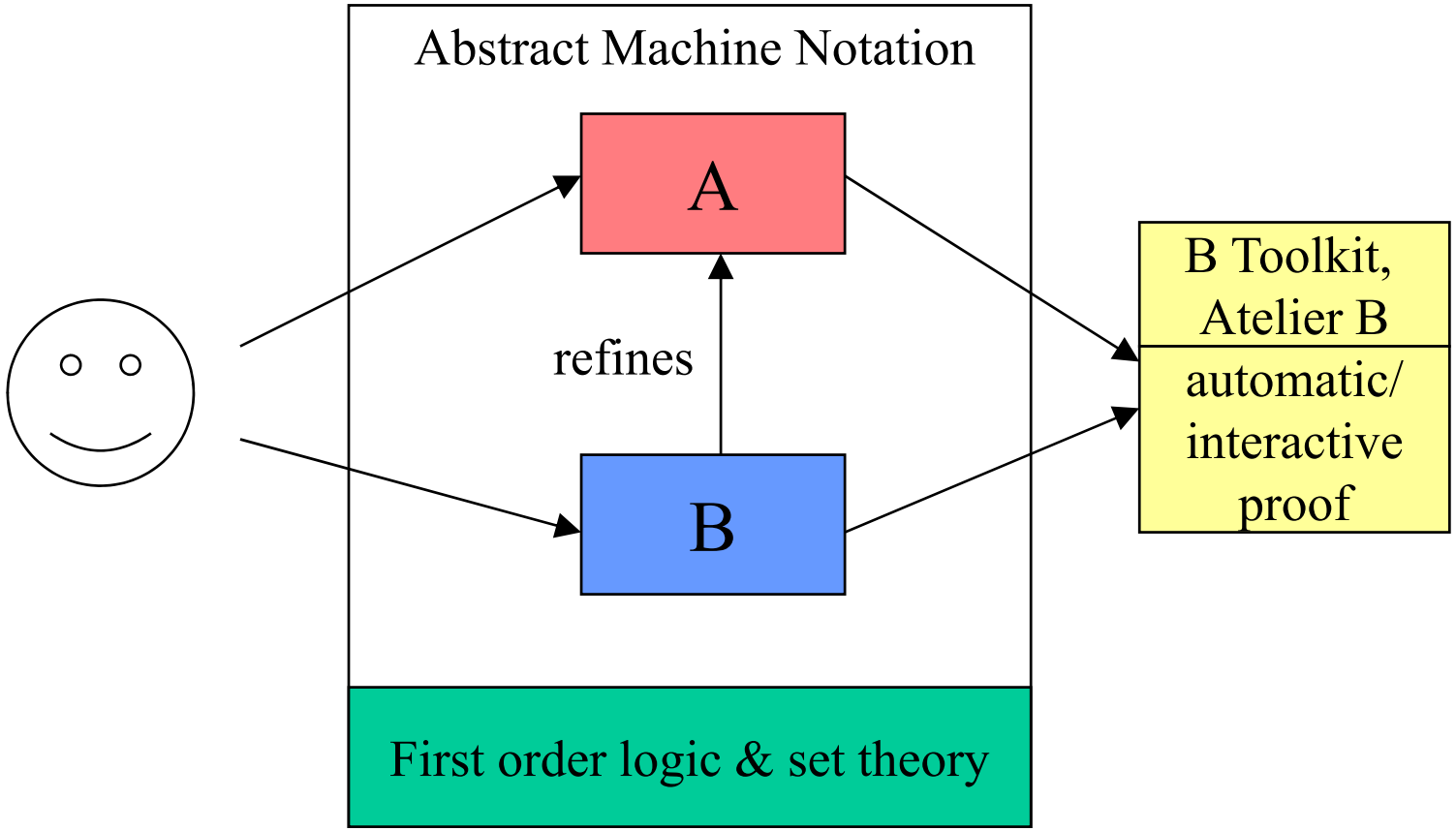


Synthesis of the Canonical Model



Canonical Model *Resource information models*

Refinement Formalization (I)



Refinement Formalization (II)

Type $U = < V_U, O_U, I_U >$ is a **refinement** of type $T = < V_T, O_T, I_T >$ if

- there exists one-to-one correspondence *Ops* between O_T and O_V;
- there exists an abstraction function $Abs: V_U \rightarrow V_T$;
- for every *o* in O_T there exists an operation *Ops(o)* = *o'* in O_U such that *o'* is a refinement of *o*:
  - precondition *pre(o)* imply precondition *pre(o')*;
  - postcondition *post(o')* imply precondition *post(o)*;

Abstract Machine Notation

- Based on first order predicate logic and Zermelo-Frenkel set theory with axiom of choice;
- allows to consider specifications of state space and behaviour in an integrated way;
- state is introduced by *state variables* together with *invariants*;
- behaviour is introduced by *operations* defined as generalized substitutions – predicate transformers;
- refinement is formalized by formulating *proof obligations*.

Refinement Formalization in AMN

REFINEMENT M
REFINES K
CONSTANTS c_M
PROPERTIES P_M
VARIABLES v
INVARIANT I_M
INITIALISATION $Init_M$
OPERATIONS
$y \leftarrow op(x) =$
PRE $Pre_{op,M}$
THEN
$Def_{op,M}$
END

REFINEMENT N
REFINES M
CONSTANTS c_N
PROPERTIES P_N
VARIABLES w
INVARIANT I_N
INITIALISATION $Init_N$
OPERATIONS
$y \leftarrow op(x) =$
PRE $Pre_{op,N}$
THEN
$Def_{op,N}$
END

Theorem of joint state non-emptiness

$$P_M \wedge P_N \Rightarrow \exists\ \mathbf{(v,w)}\ (I_M \wedge I_N)$$

Theorem of initialization refinement

$$P_M \wedge P_N \Rightarrow [Init_N] \neg [Init_M] \neg I_N$$

Theorem of operation refinement

$$P_M \wedge P_N \wedge I_M \wedge I_N \wedge Pre_{op,M} \Rightarrow Pre_{op,N} \wedge [Def_{op,N}\{y \rightarrow y'\}] \neg [Def_{op,M}] \neg (I_N \wedge y'=y)$$

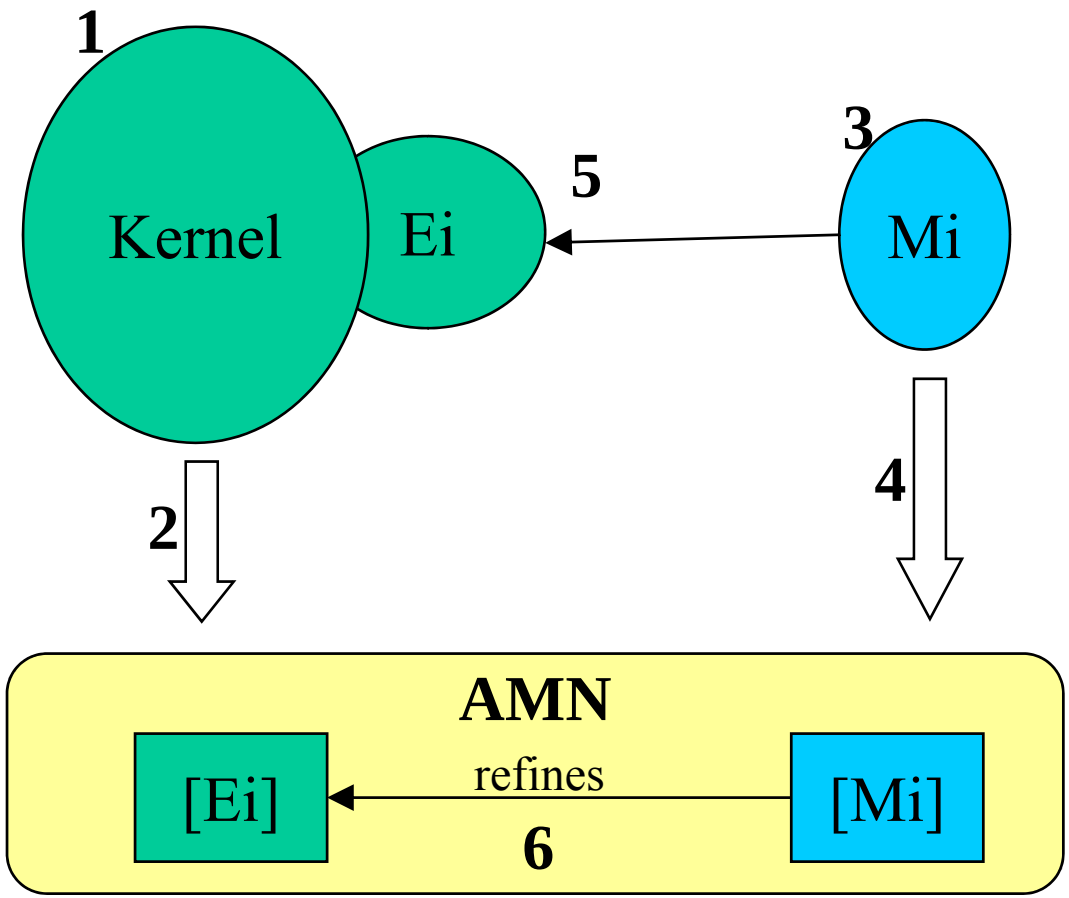
"Operation refinement"

Under the refinement relation and the precondition of the more abstract operation, the precondition of the more concrete operation holds;

For every execution of concrete operation there is a corresponding execution of abstract operation from the same initial state which establishes the same external result values and reestablishes the refinement relation between the post-states.

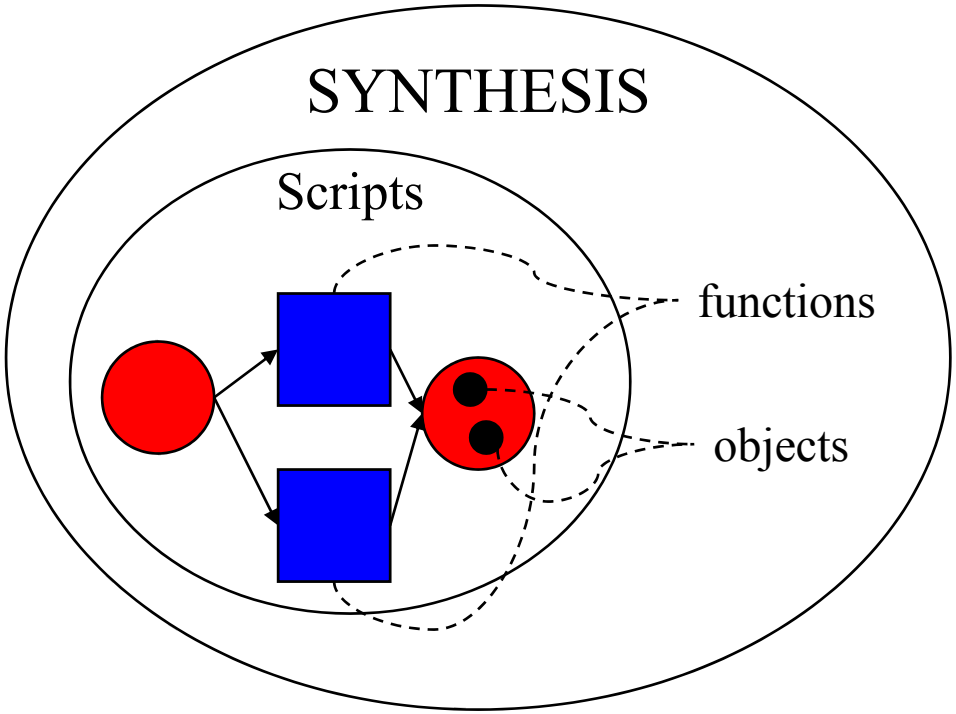
Main Points of the Canonical Process Model (CPM) Synthesis

1. CPM kernel
2. AMN-semantics of kernel
3. source models Mi
4. AMN-semantics of source models
5. extensions Ei and mapping Mi → Ei
6. [Mi] refines [Ei]



Kernel of the Canonical Process Model

- subset of *scripts* of the SYTHESIS language
- based on Petri Nets model
- tokens are objects
- transitions are binded with functions

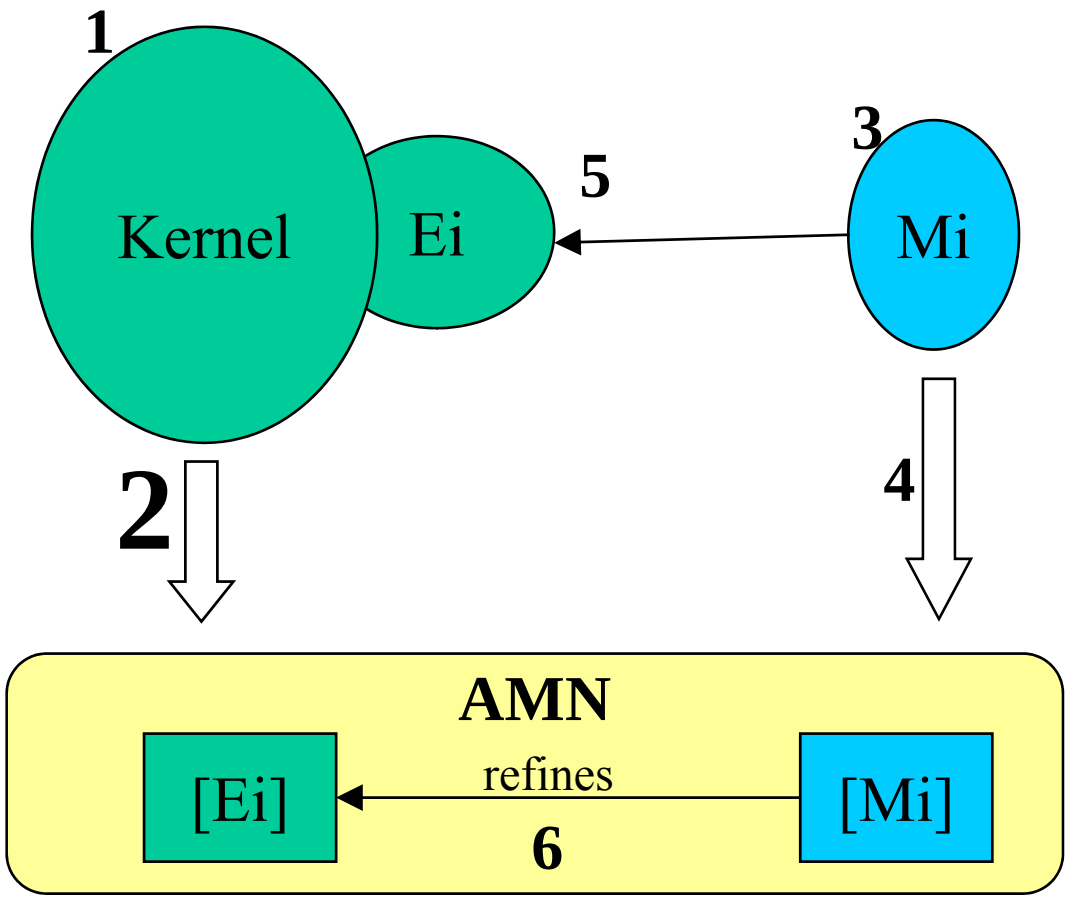


Example of a Script

```
{ discriminator; in: script;
  params: { branch1/function, … , entrance1TokenType/type, … };
  states: {entrance1; token: entrance1TokenType;}, …
  transitions:
   {Branch1;
    from: entrance1; bind_from: {entrance1, in};
    to: auxPlace1; bind_to: {auxPlace1, out};
    activity: {in: function;
     params: {+in/entrance1TokenType, -out/auxPlace1TokenType};
     {{branch1(in, out)}}
    }
   },
   …
}
```

Main Points of the Canonical Process Model (CPM) Synthesis

1. CPM kernel
2. AMN-semantics of kernel
3. source models Mi
4. AMN-semantics of source models
5. extensions Ei and mapping Mi → Ei
6. [Mi] refines [Ei]



AMN-semantics of the CPM Kernel

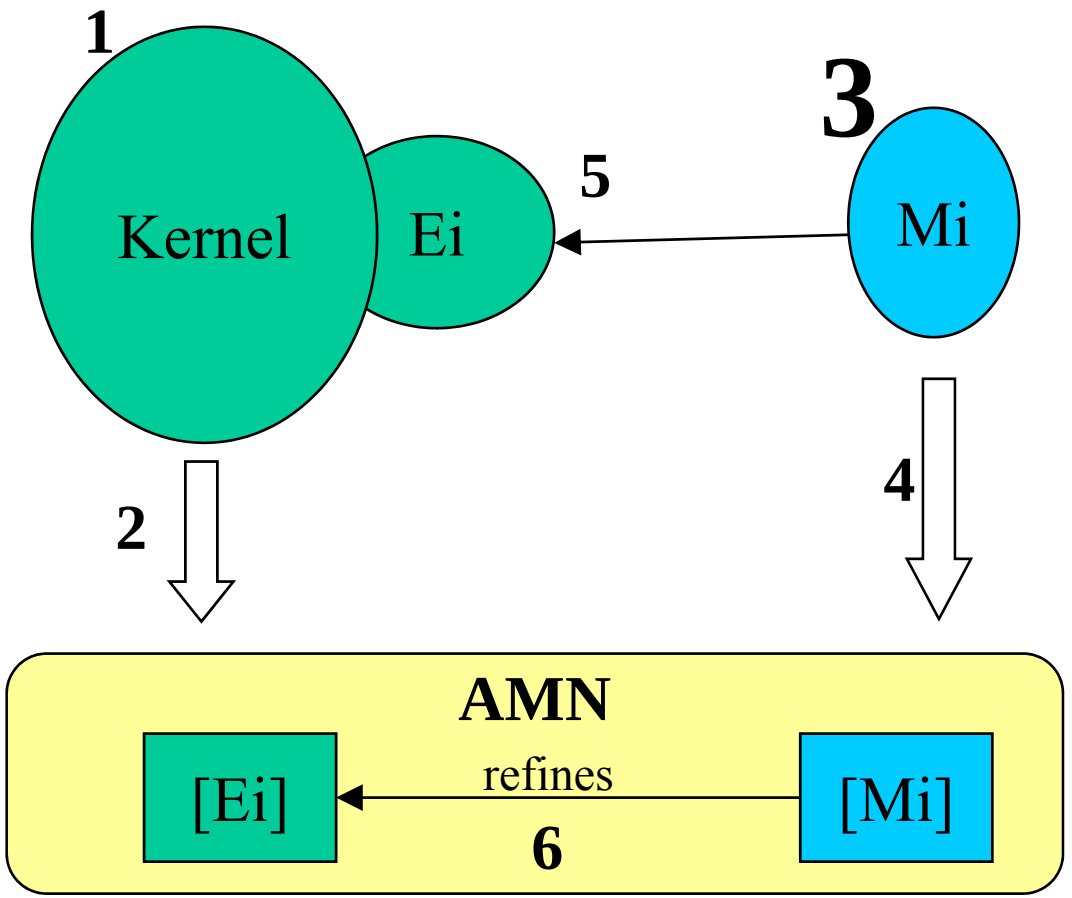
- [Abrial] – Event B
- [Butler] – csp2B: A Practical Approach to Combining CSP and B
- [Treharne, Schneider] – How to Drive a B-machine
- [Butler, Snook] – Verifying Dynamic Properties of UML Models by Translation to the B language and Toolkit
- [Ledang, Souquieres] – Contributions for Modeling UML State-Charts in B

Example: AMN-semantics of the discriminator Script

REFINEMENT *DiscriminatorScript*
SETS *Obj*
CONSTANTS *ext_entrance1TokenType, ...*
PROPERTIES *ext_entrance1TokenType:* POW*(Obj) ...*
VARIABLES *entrance1, ...*
INVARIANT *entrance1:* POW*(ext_entrance1TokenType) ...*
OPERATIONS
Branch1 =
SELECT *#t.(t: entrance1)* THEN
ANY *t* WHERE *t: entrance1* THEN
entrance1:= entrance1 – {t} ||
ANY *r* WHERE *r: ext_auxState1TokenType* THEN
auxState1:= auxState1 \/ {r}
END
END
END
...

Main Points of the Canonical Process Model (CPM) Synthesis

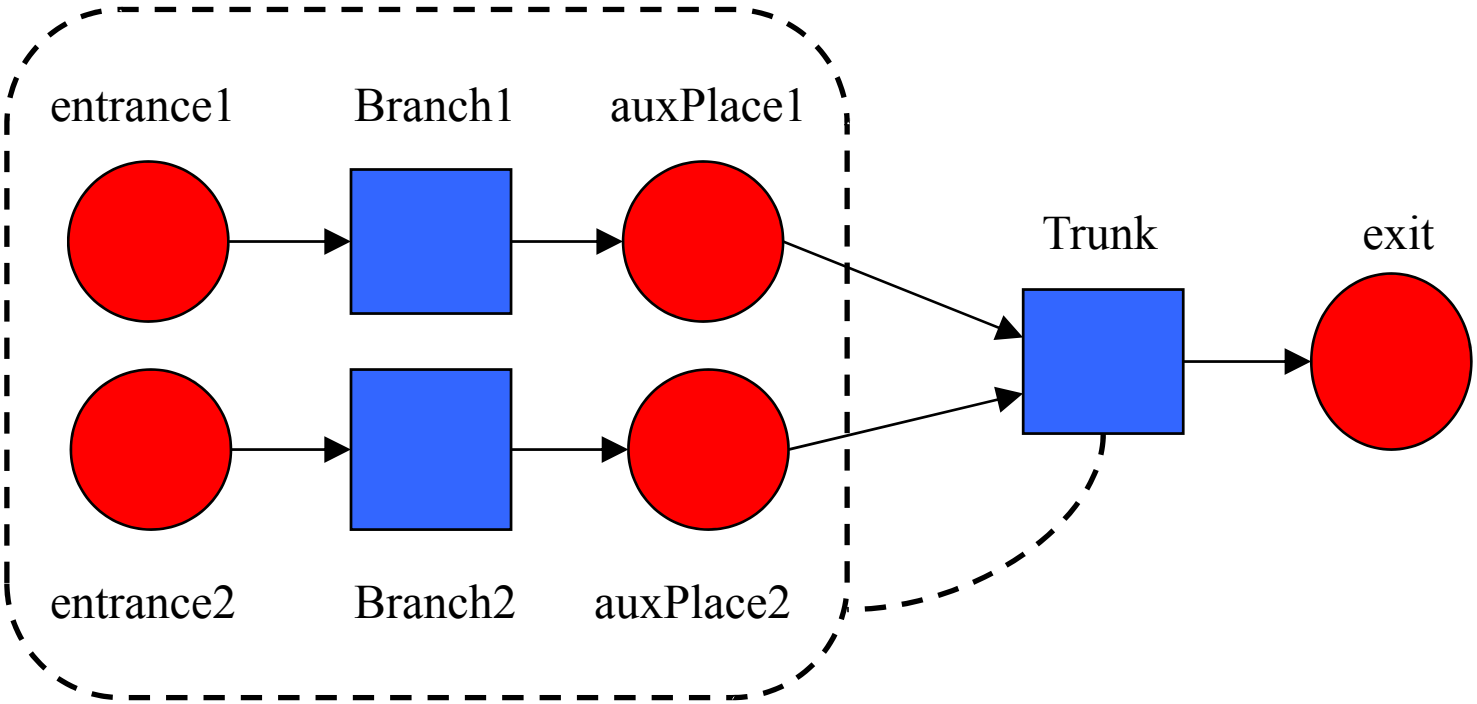
1. CPM kernel
2. AMN-semantics of kernel
3. source models Mi
4. AMN-semantics of source models
5. extensions Ei and mapping Mi → Ei
6. [Mi] refines [Ei]



Source Models

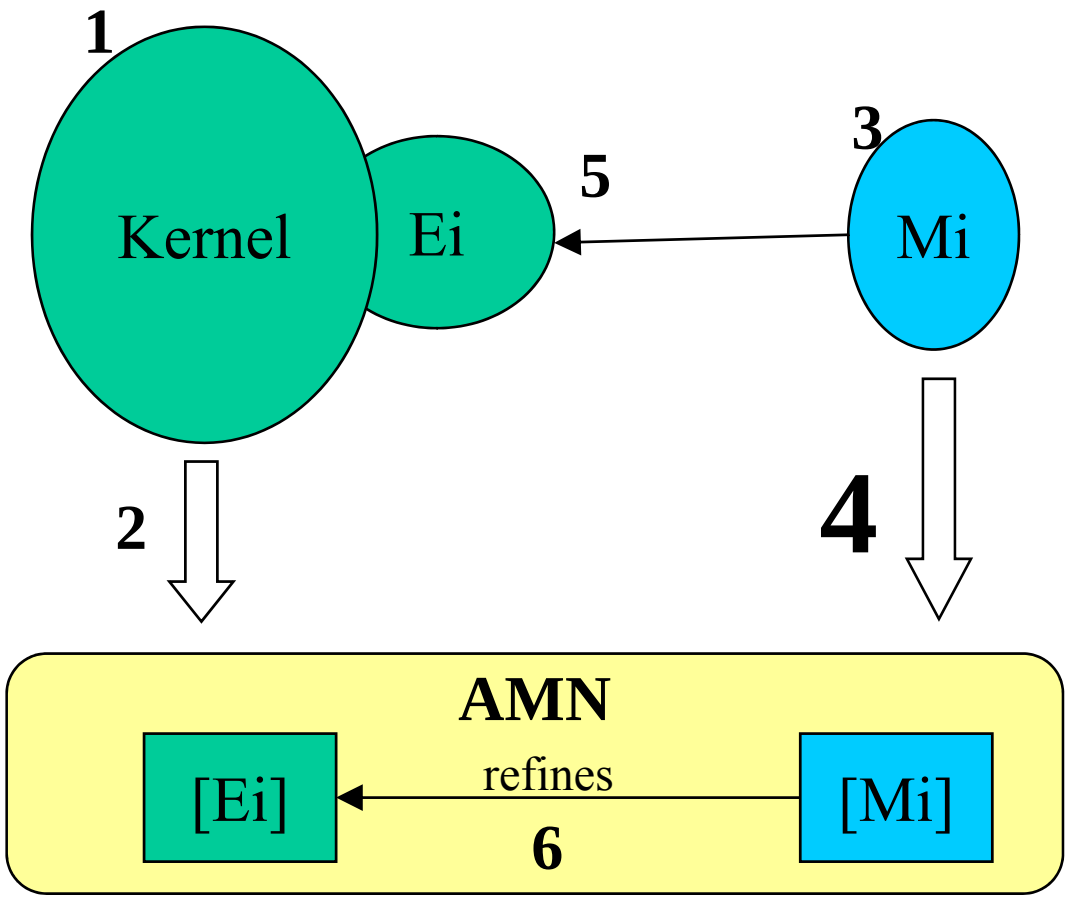
- [van der Aalst, 2003] The analysis of large number of WfMS process models;
- as a result 20 *workflow patterns* were obtained;
- set of workflow patterns is *complete*;
- every pattern is considered as a source model.

An Example of Workflow Pattern: Discriminator



Main Points of the Canonical Process Model (CPM) Synthesis

1. CPM kernel
2. AMN-semantics of kernel
3. source models Mi
4. AMN-semantics of source models
5. extensions Ei and mapping Mi → Ei
6. [Mi] refines [Ei]



AMN-semantics of Source Models

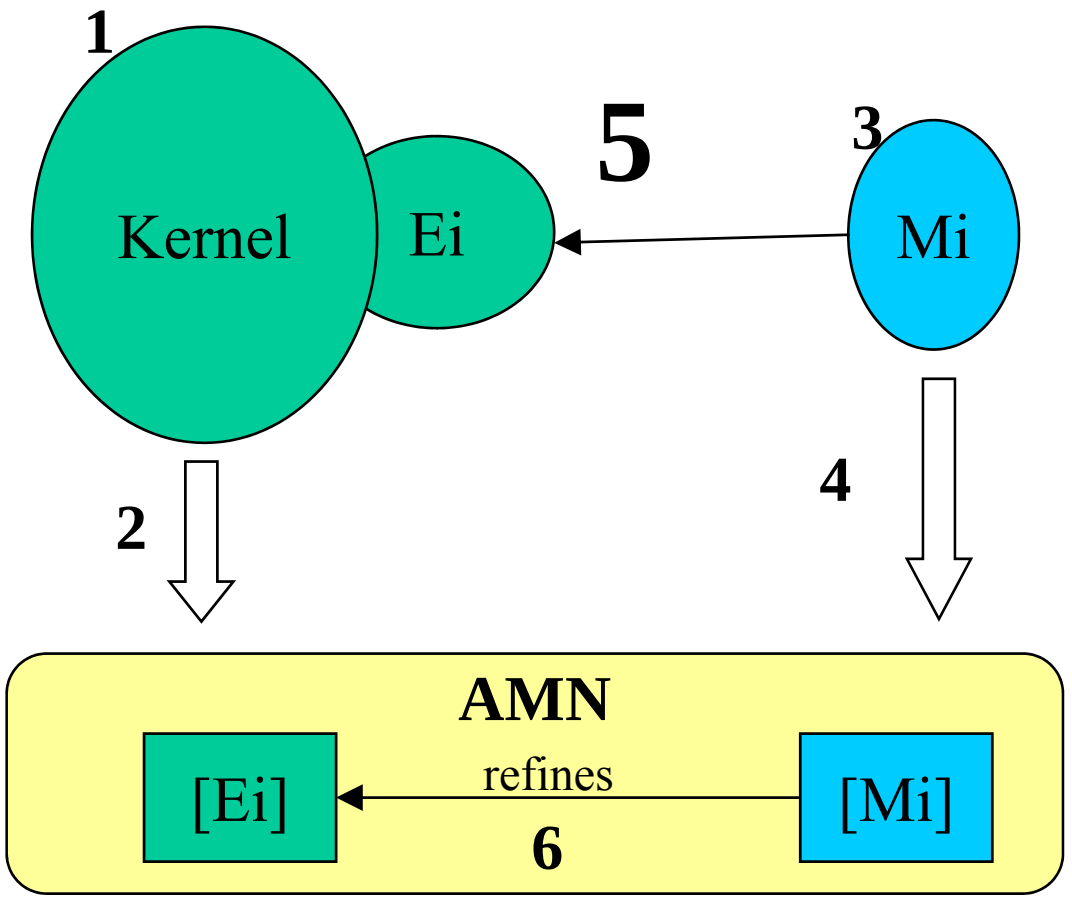
- Workflow patterns are defined in YAWL (Yet Another Workflow Language) developed by van der Aalst;
- workflow specification in YAWL is a set of Extended Workflow Nets (EWF-nets), forming a hierarchical tree-like structure;
- EWF-net is a tuple <C, i, o, T, F, join, split, rem>;
- an appropriate AMN-semantics was defined for YAWL.

Example: AMN-semantics of the Discriminator Pattern

REFINEMENT *Discriminator*
SETS *States = { state_enter1, ...}*
VARIABLES *states, ...*
INVARIANT *states: States* → NAT ...
OPERATIONS
enter_branch1 =
SELECT *exec_branch1=0 & states(state_enter1)>0*
THEN
ANY *t* WHERE *t: entrance1* THEN
states(state_enter1):= states(state_enter1)-1 ||
exec_branch1:= exec_branch1+1
END
END
...

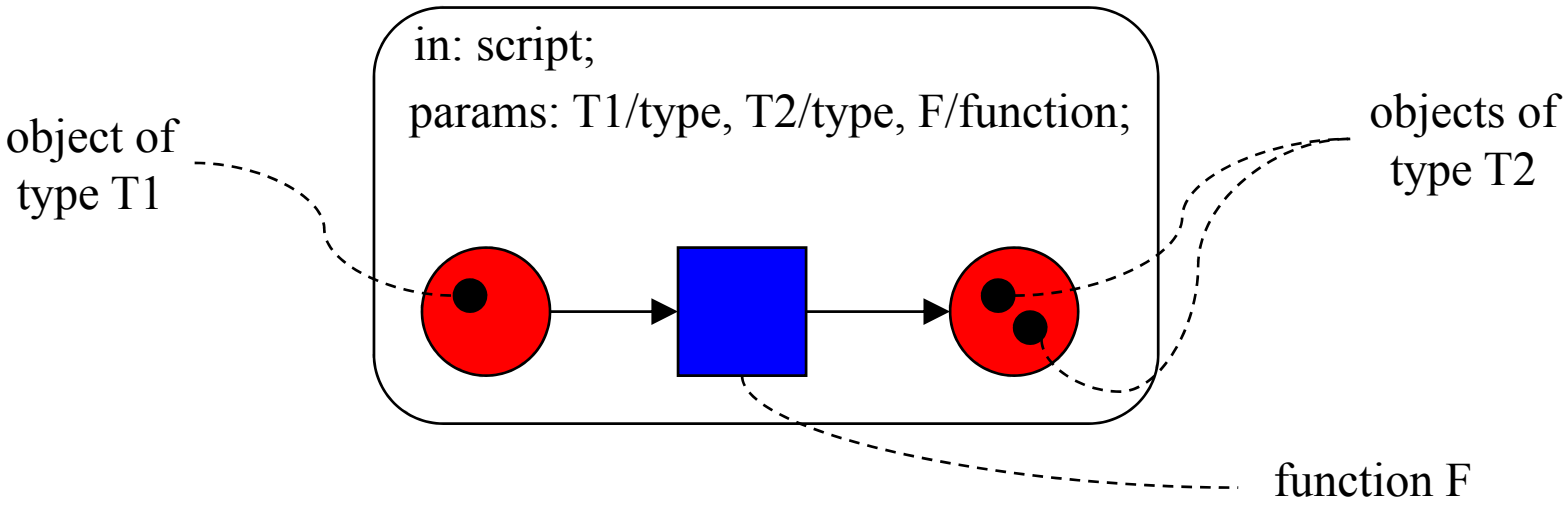
Main Points of the Canonical Process Model (CPM) Synthesis

1. CPM kernel
2. AMN-semantics of kernel
3. source models Mi
4. AMN-semantics of source models
5. extensions Ei and mapping Mi → Ei
6. [Mi] refines [Ei]



Extensions of the Kernel

- For every workflow pattern Mi a kernel extension Ei is a generic (parameterized) script type;
- parameters of a script type are *types of the places* and
- *functions binded with the transitions*.

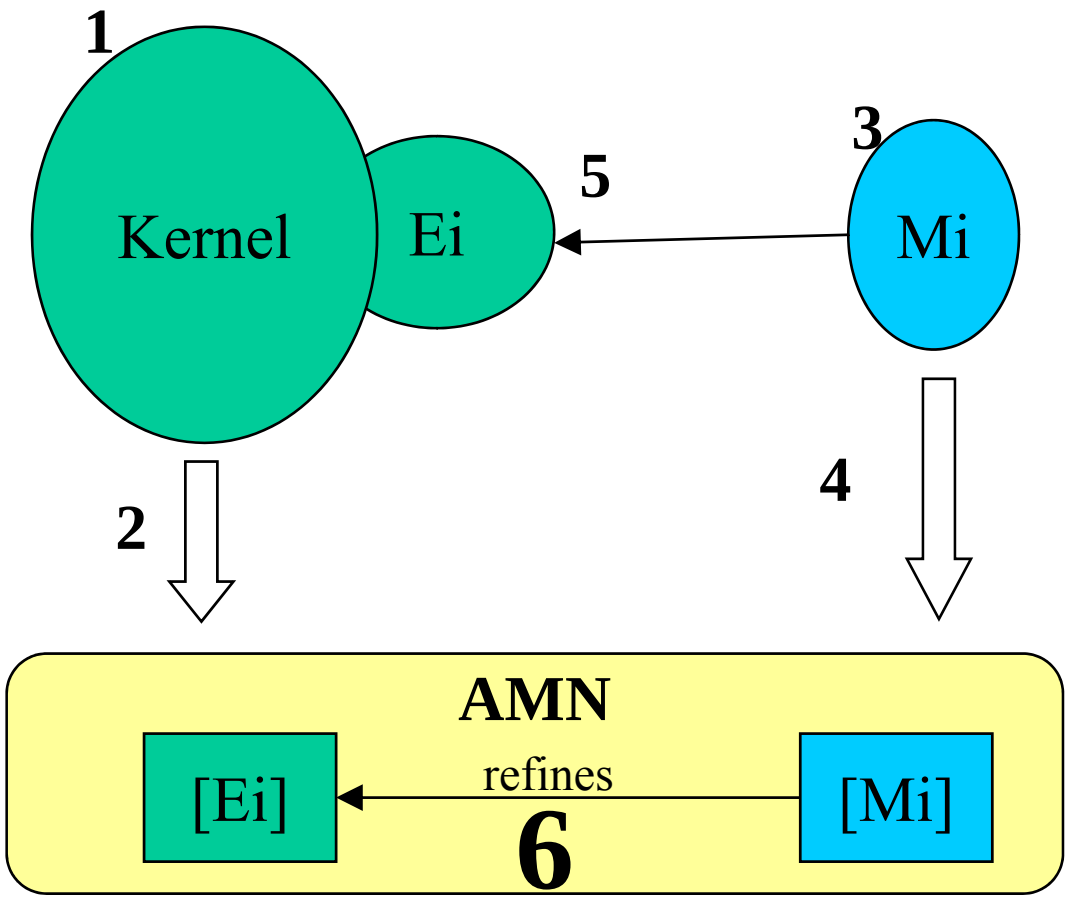


Extension of the Kernel by Discriminator Script

```
{ discriminator; in: script;
  params: { branch1/function, … , entrance1TokenType/type, … };
  states: {entrance1; token: entrance1TokenType;}, …
  transitions:
   {Branch1;
    from: entrance1; bind_from: {entrance1, in};
    to: auxPlace1; bind_to: {auxPlace1, out};
    activity: {in: function;
     params: {+in/entrance1TokenType, -out/auxPlace1TokenType};
     {{branch1(in, out)}}
    }
   },
   …
}
```

Main Points of the Canonical Process Model (CPM) Synthesis

1. CPM kernel
2. AMN-semantics of kernel
3. source models Mi
4. AMN-semantics of source models
5. extensions Ei and mapping Mi → Ei
6. [Mi] refines [Ei]



Statistics for the Proof of Refining Discriminator Script Type by Discriminator Pattern

Kind of theorem	Number of theorems	Number of automatically proved theorems
The theorem of the unified state non-emptiness	1	0
Theorems of the initialisation refinement	6	6
Theorems of refinement for operation *enter_branch1*	7	5
Theorems of refinement for operation *exit_branch1*	7	4
Theorems of refinement for operation *enter_branch2*	7	5
Theorems of refinement for operation *exit_branch2*	8	5
Theorems of refinement for operation *enter_trunk*	16	11
Theorems of refinement for operation *exit_trunk*	13	2
Total number of theorems	65	38

Conclusions

- Canonical process model was synthesized: kernel was chosen and extensions corresponding to 20 workflow patterns were defined;
- the process of extension was formally verified;
- the canonical process model can be used as a basis for the methods of integration, reuse and composition of the heterogeneous process components.