

# AmbientTalk

A scripting language for mobile phones

Tom Van Cutsem

# Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication



# Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication

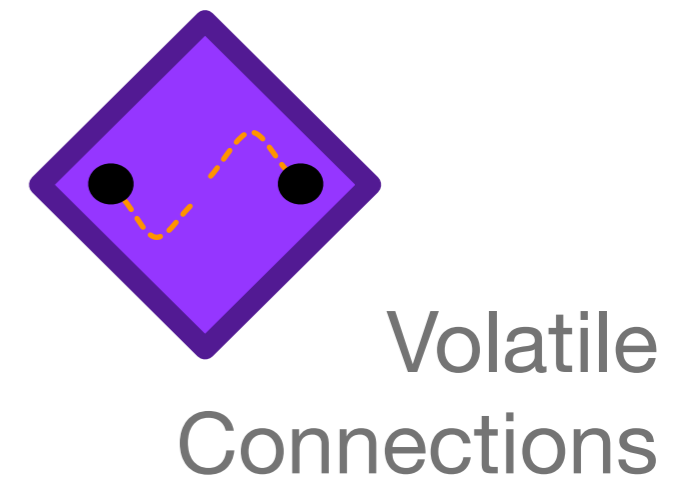
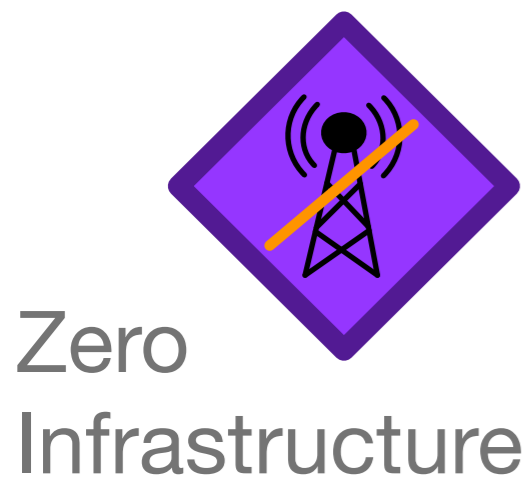


Zero  
Infrastructure



# Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication




# Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication



# AmbientTalk: fact sheet

---

- Object-oriented, functional patterns, dynamically typed
- Actor-based concurrency/distribution
- Mirror-based reflection
- JVM as platform
- Runs on  and J2ME/CDC phones



# Four Decades of Language Research



Smalltalk  
'70s-'80s



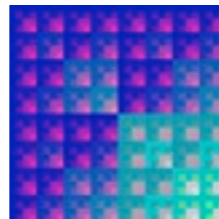
1986



Scheme  
1975



1997

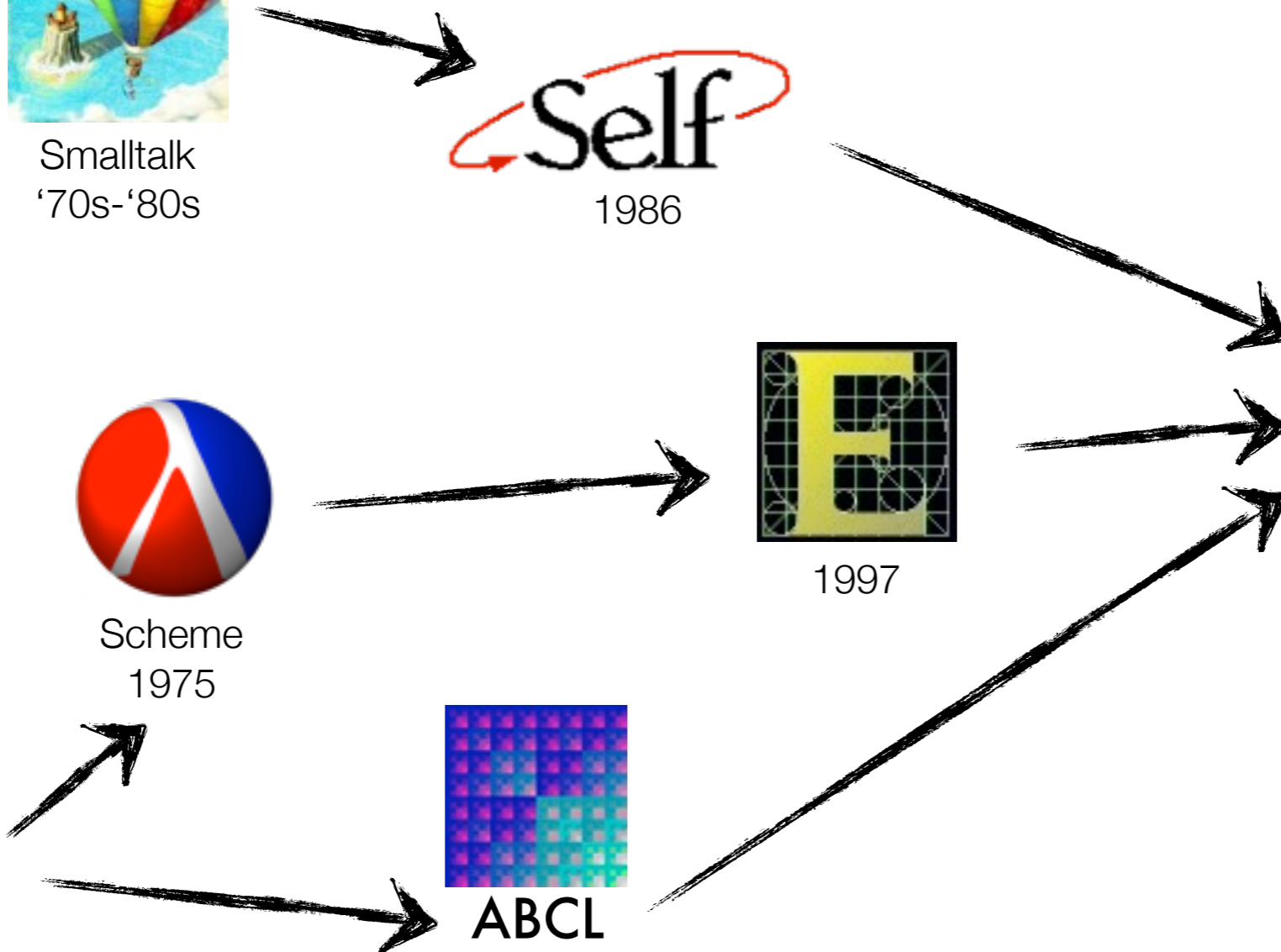


ABCL  
1986



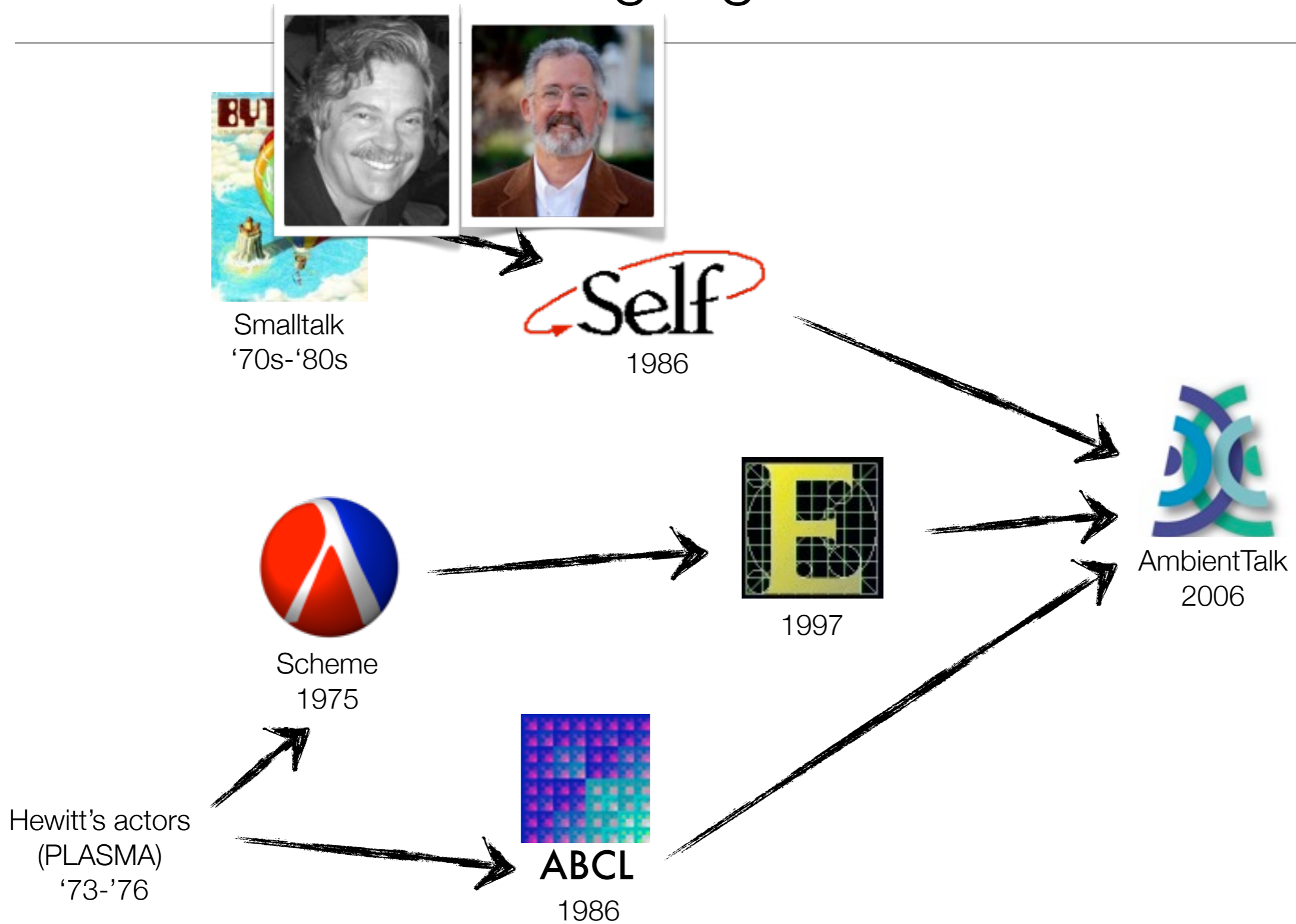
AmbientTalk  
2006

Hewitt's actors  
(PLASMA)  
'73-'76





# Four Decades of Language Research





# Four Decades of Language Research



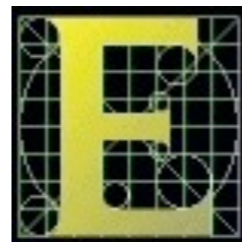
Smalltalk  
'70s-'80s



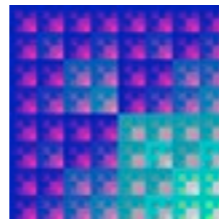
1986



Scheme  
1975



1997

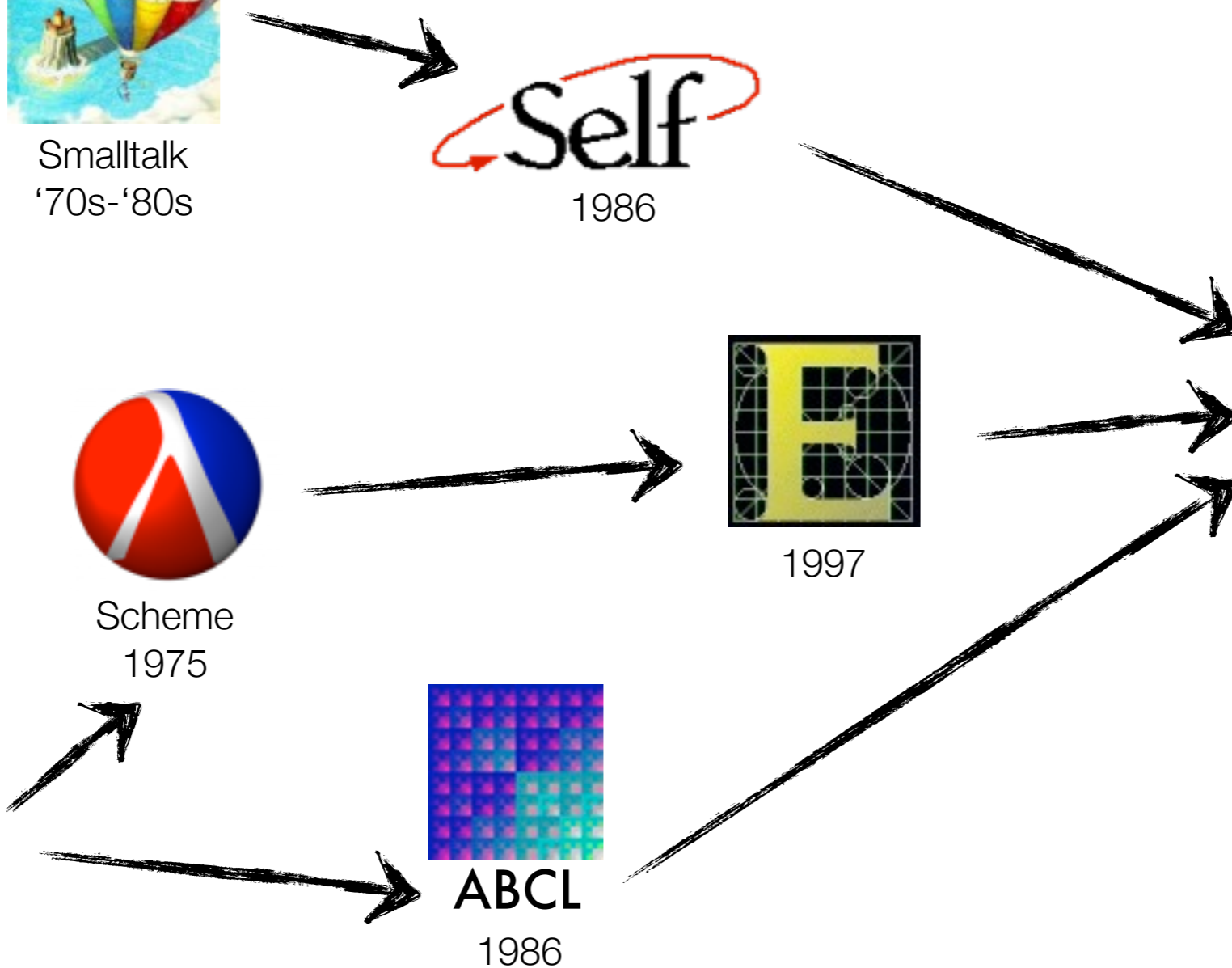


ABCL  
1986

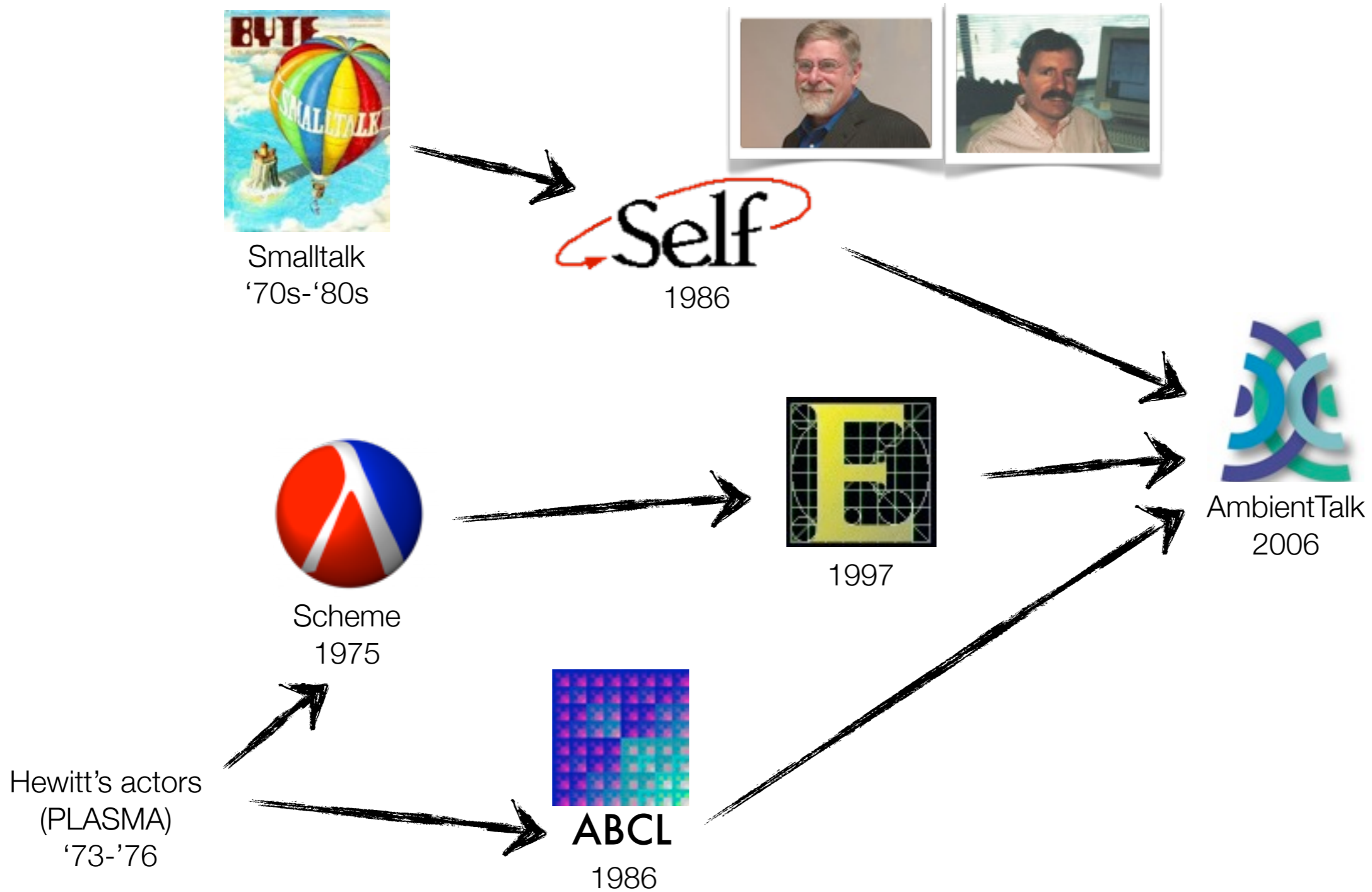


AmbientTalk  
2006

Hewitt's actors  
(PLASMA)  
'73-'76



# Four Decades of Language Research



# Four Decades of Language Research



Smalltalk  
'70s-'80s



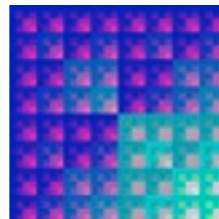
1986



Scheme  
1975



1997

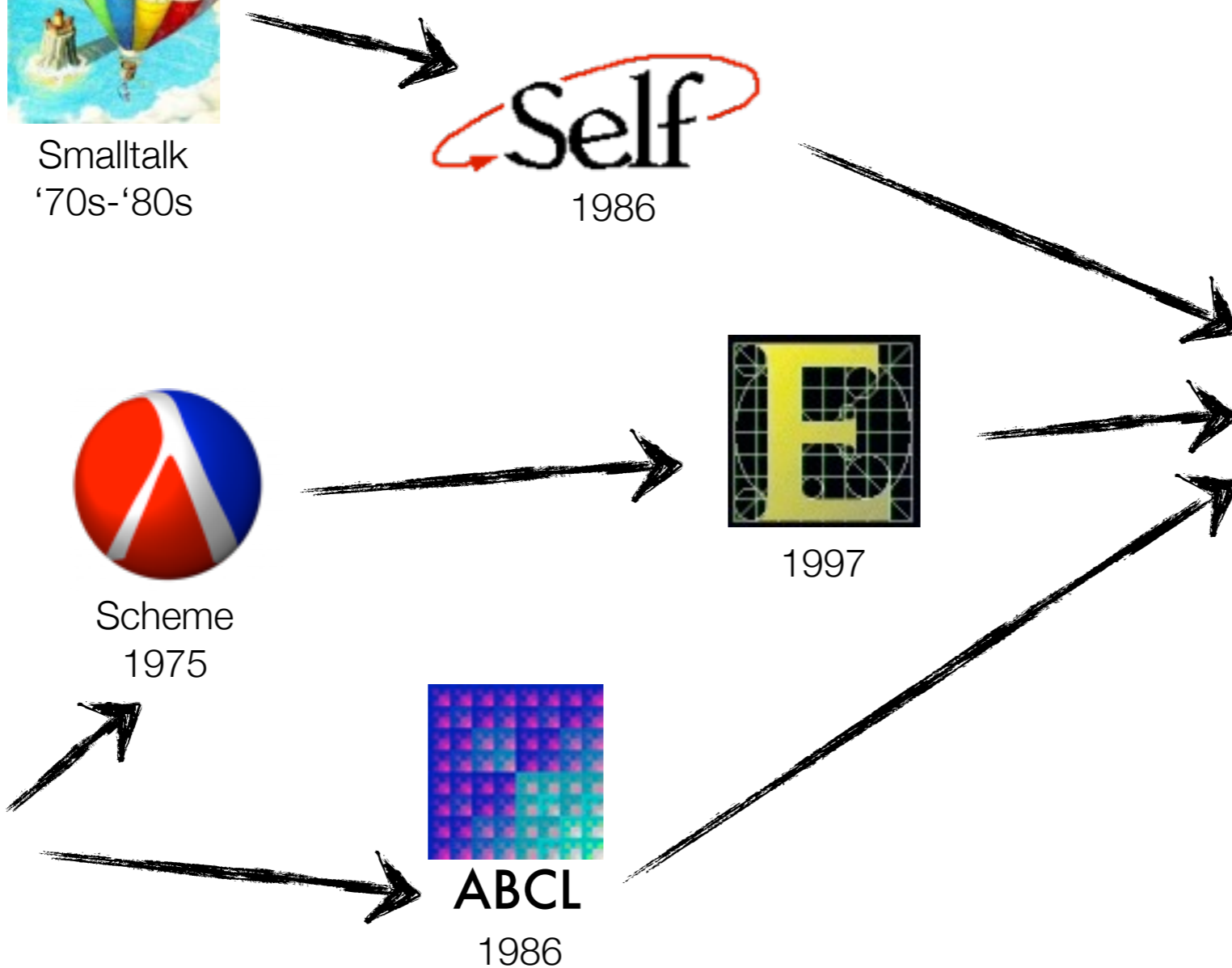


ABCL  
1986



AmbientTalk  
2006

Hewitt's actors  
(PLASMA)  
'73-'76



# Four Decades of Language Research



Smalltalk  
'70s-'80s



1986



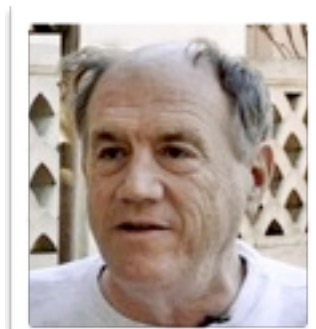
Scheme  
1975



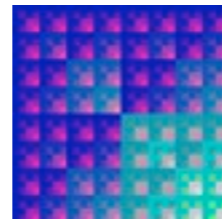
1997



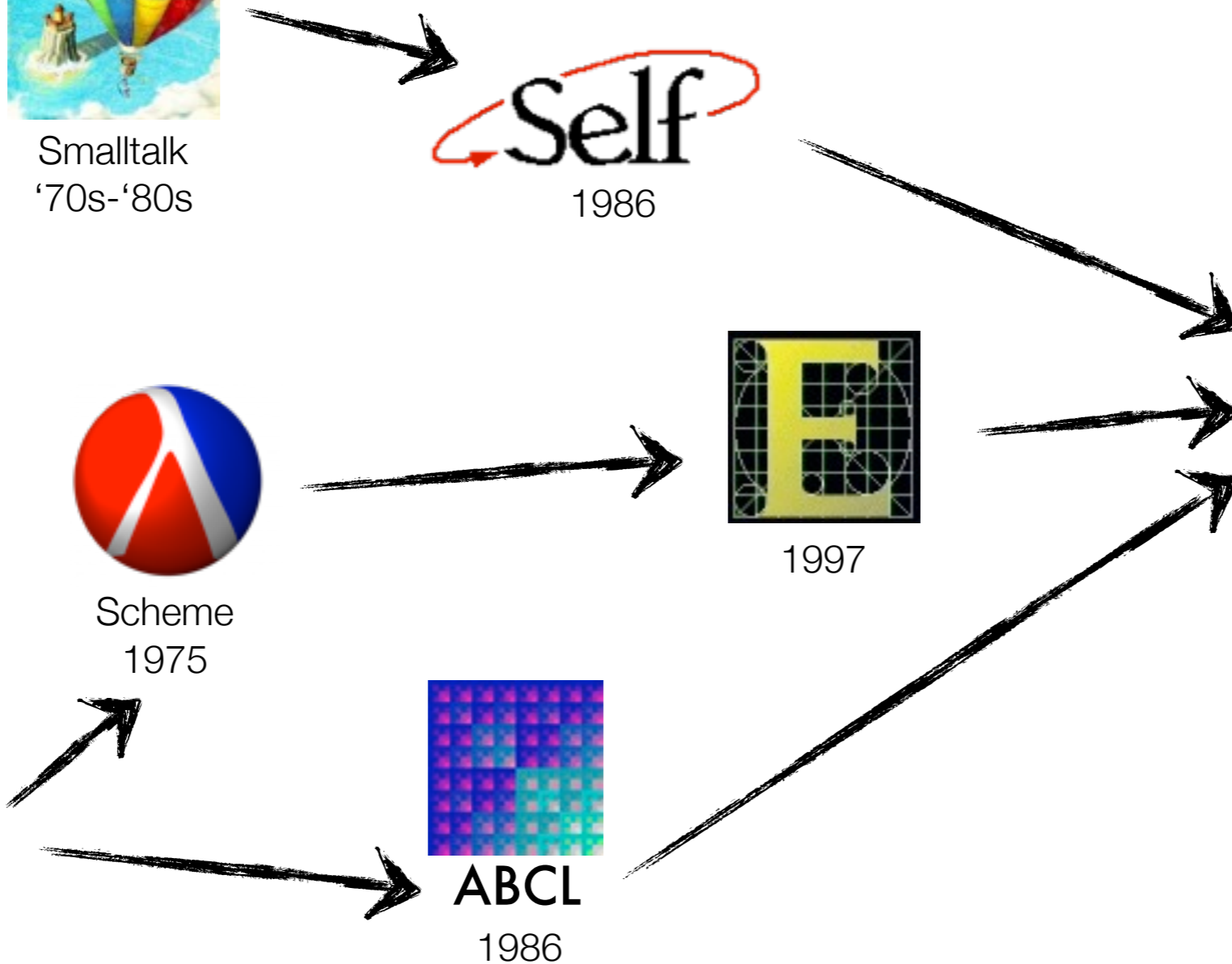
AmbientTalk  
2006



Hewitt's actors  
(PLASMA)  
'73-'76



ABCL  
1986



# Four Decades of Language Research



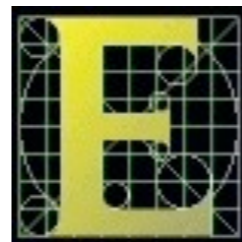
Smalltalk  
'70s-'80s



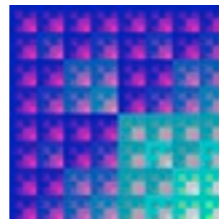
1986



Scheme  
1975



1997

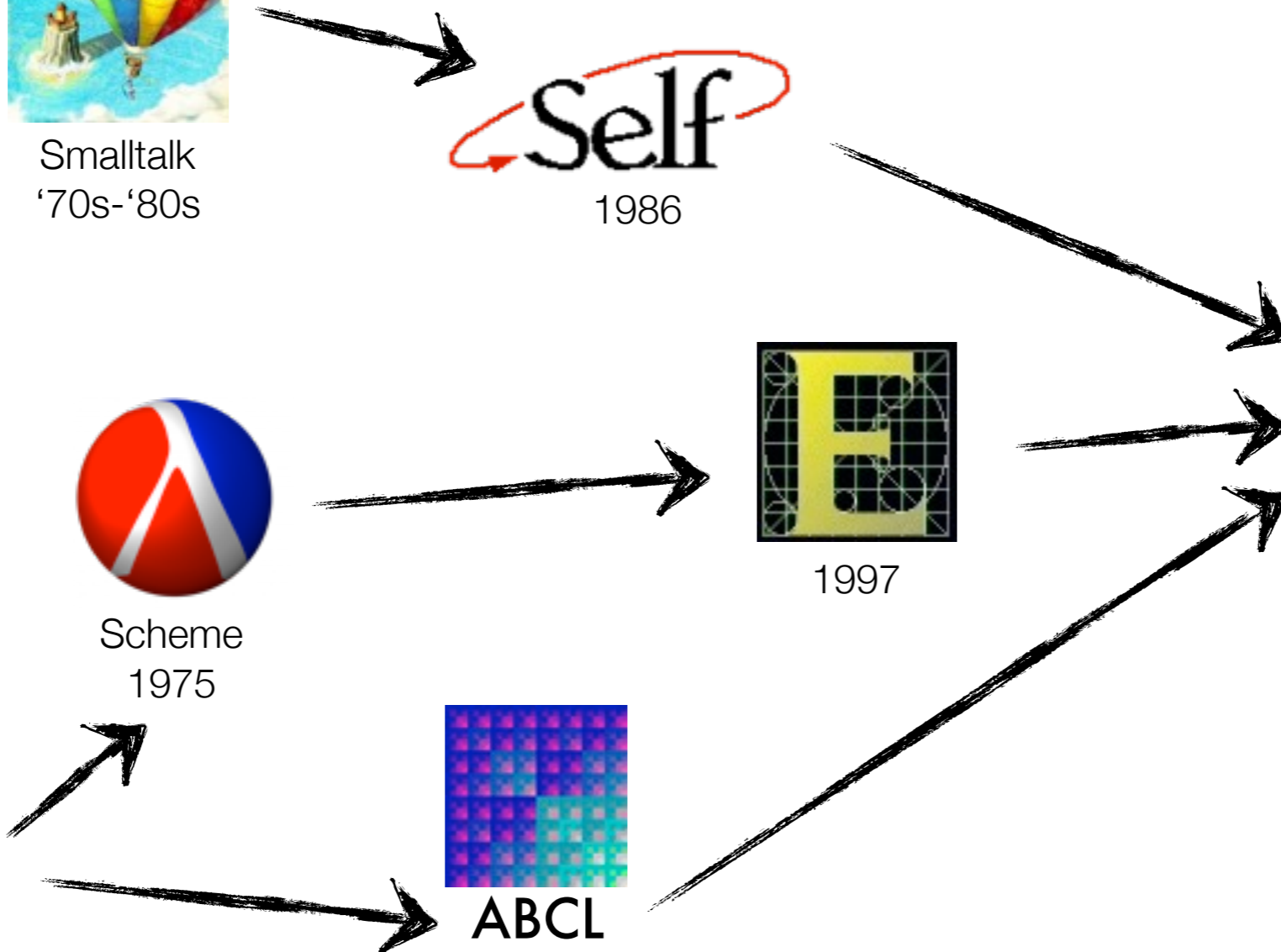


ABCL  
1986



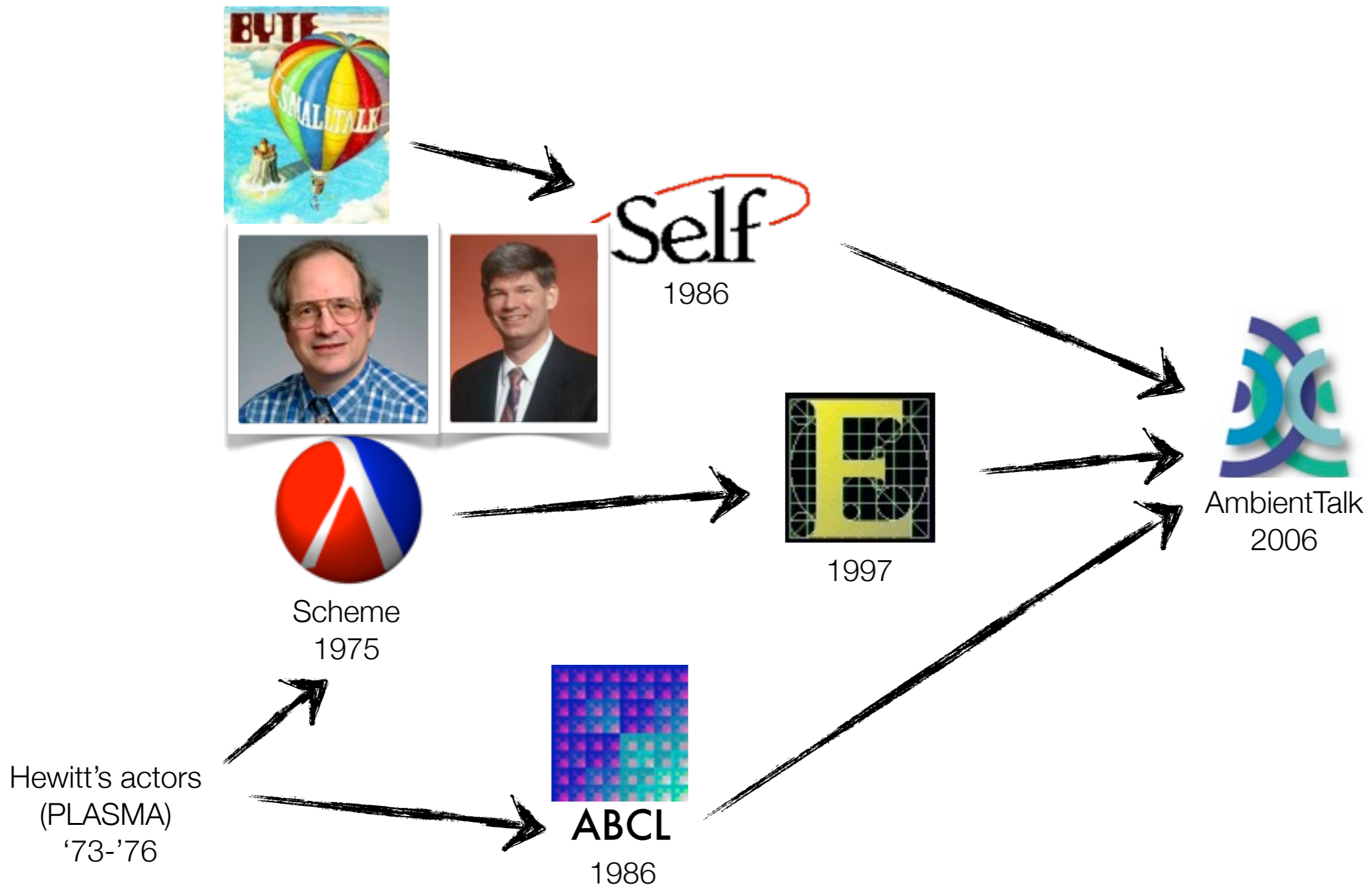
AmbientTalk  
2006

Hewitt's actors  
(PLASMA)  
'73-'76





# Four Decades of Language Research



# Four Decades of Language Research



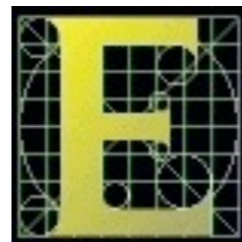
Smalltalk  
'70s-'80s



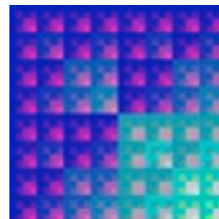
1986



Scheme  
1975



1997

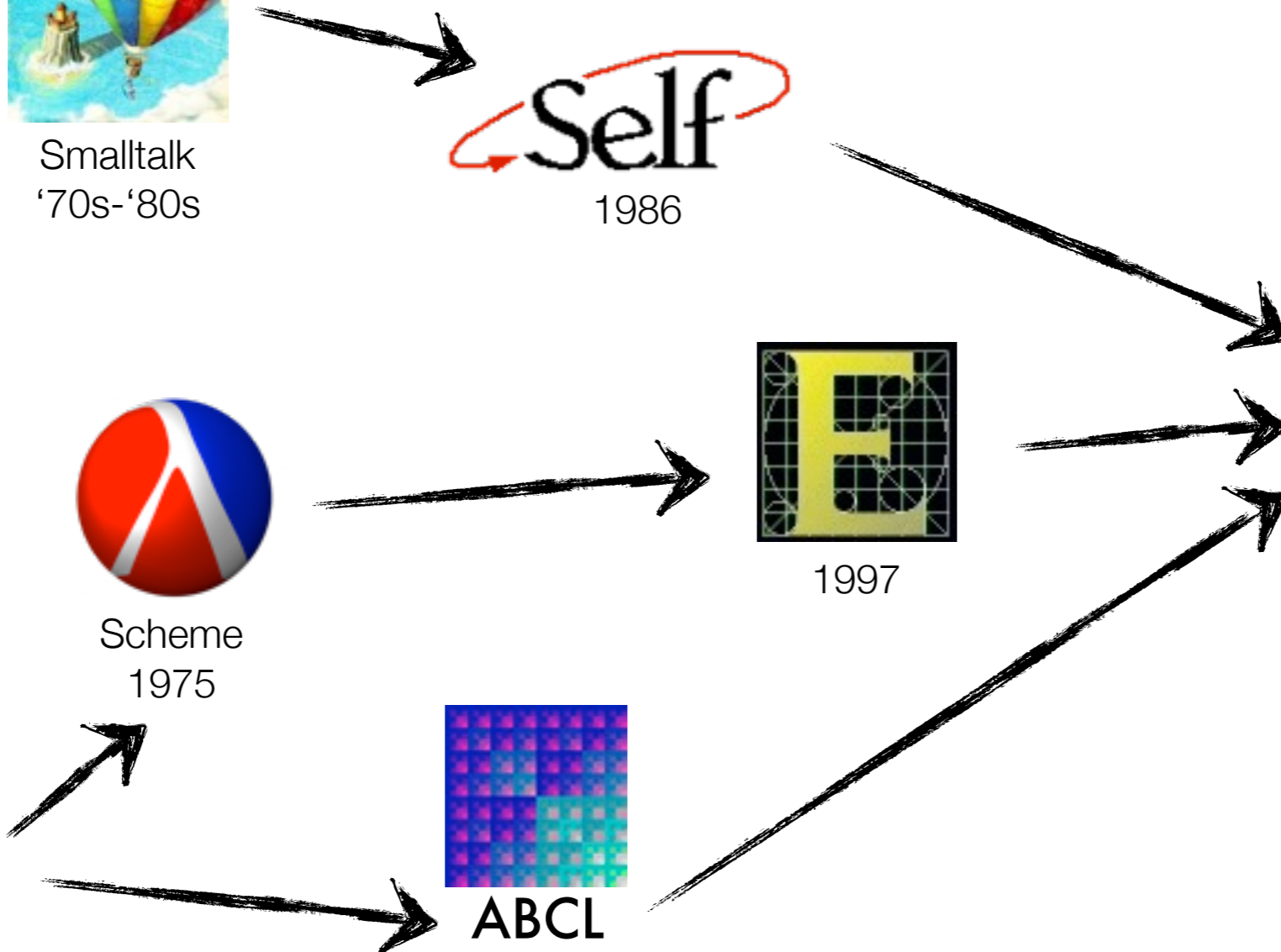


ABCL  
1986



AmbientTalk  
2006

Hewitt's actors  
(PLASMA)  
'73-'76





# Four Decades of Language Research



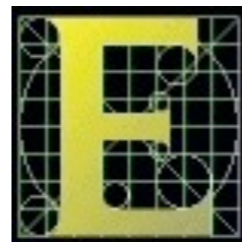
Smalltalk  
'70s-'80s



1986

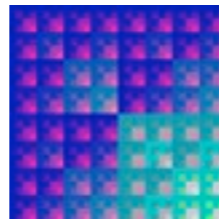


Scheme  
1975



1997

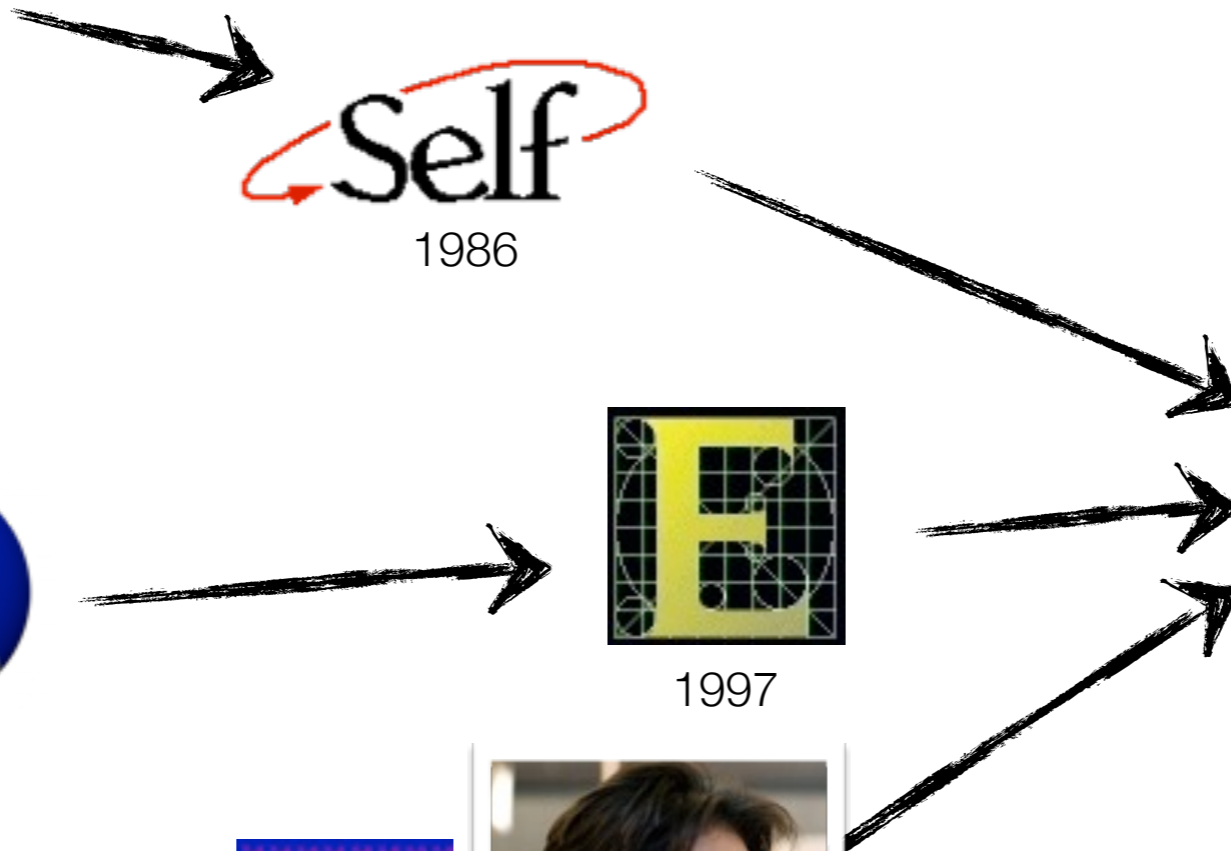
Hewitt's actors  
(PLASMA)  
'73-'76



ABCL  
1986



AmbientTalk  
2006



# Four Decades of Language Research



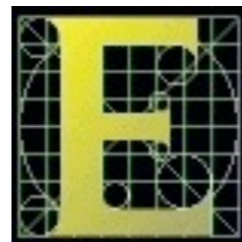
Smalltalk  
'70s-'80s



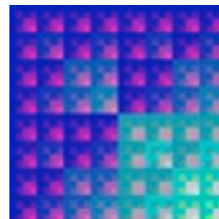
1986



Scheme  
1975



1997

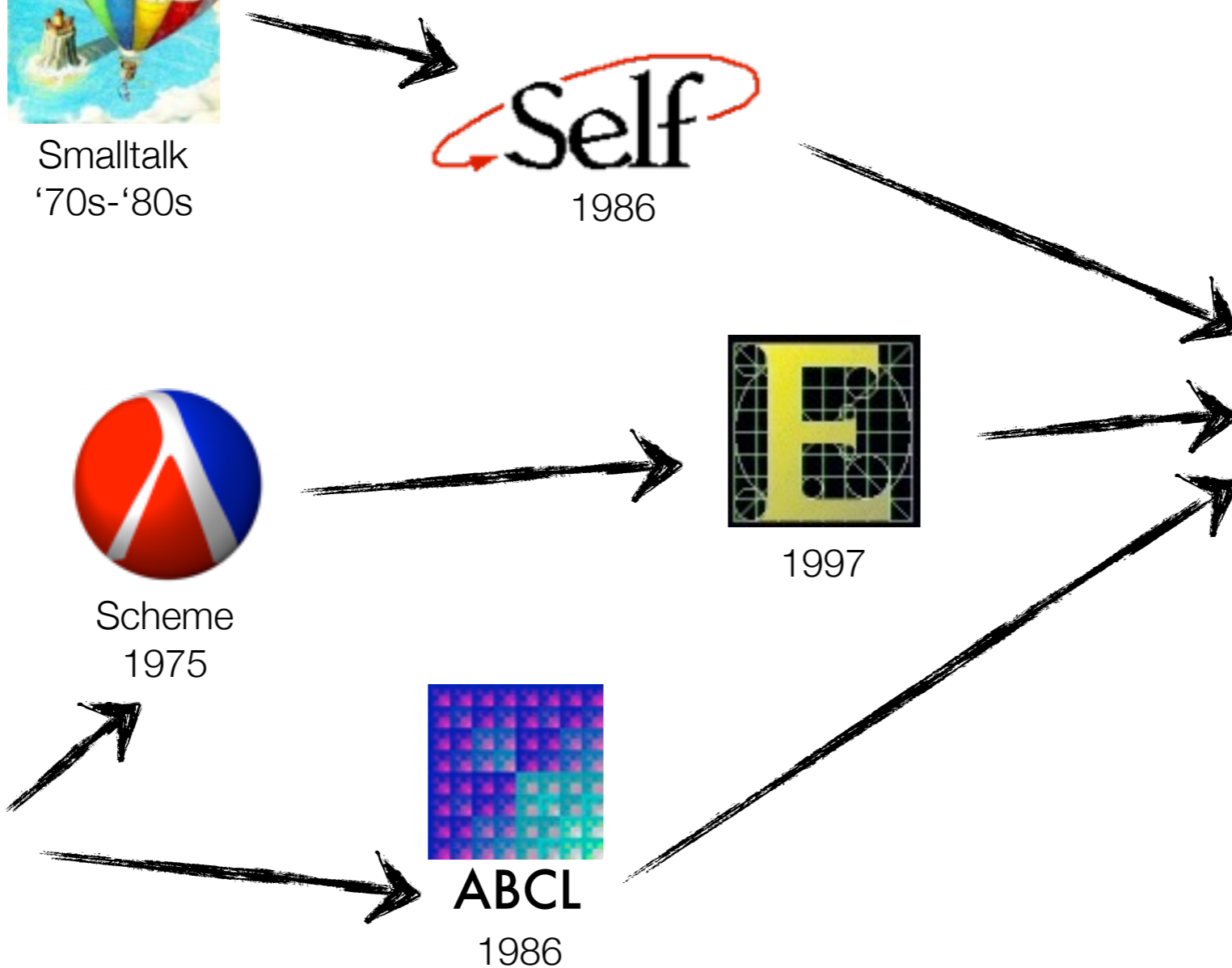


ABCL  
1986

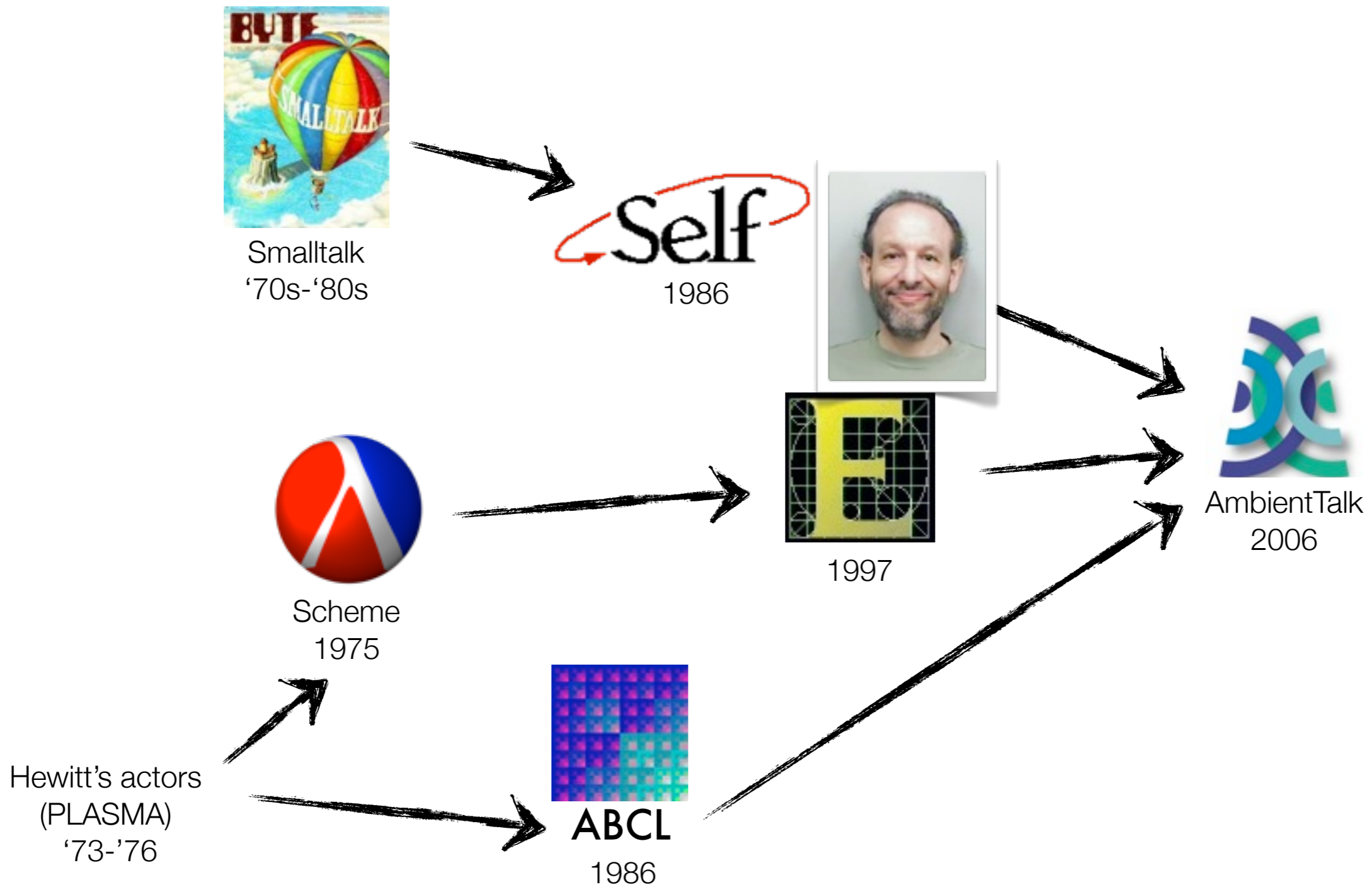


AmbientTalk  
2006

Hewitt's actors  
(PLASMA)  
'73-'76



# Four Decades of Language Research



# Four Decades of Language Research



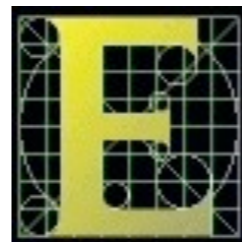
Smalltalk  
'70s-'80s



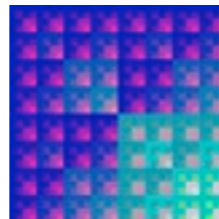
1986



Scheme  
1975



1997

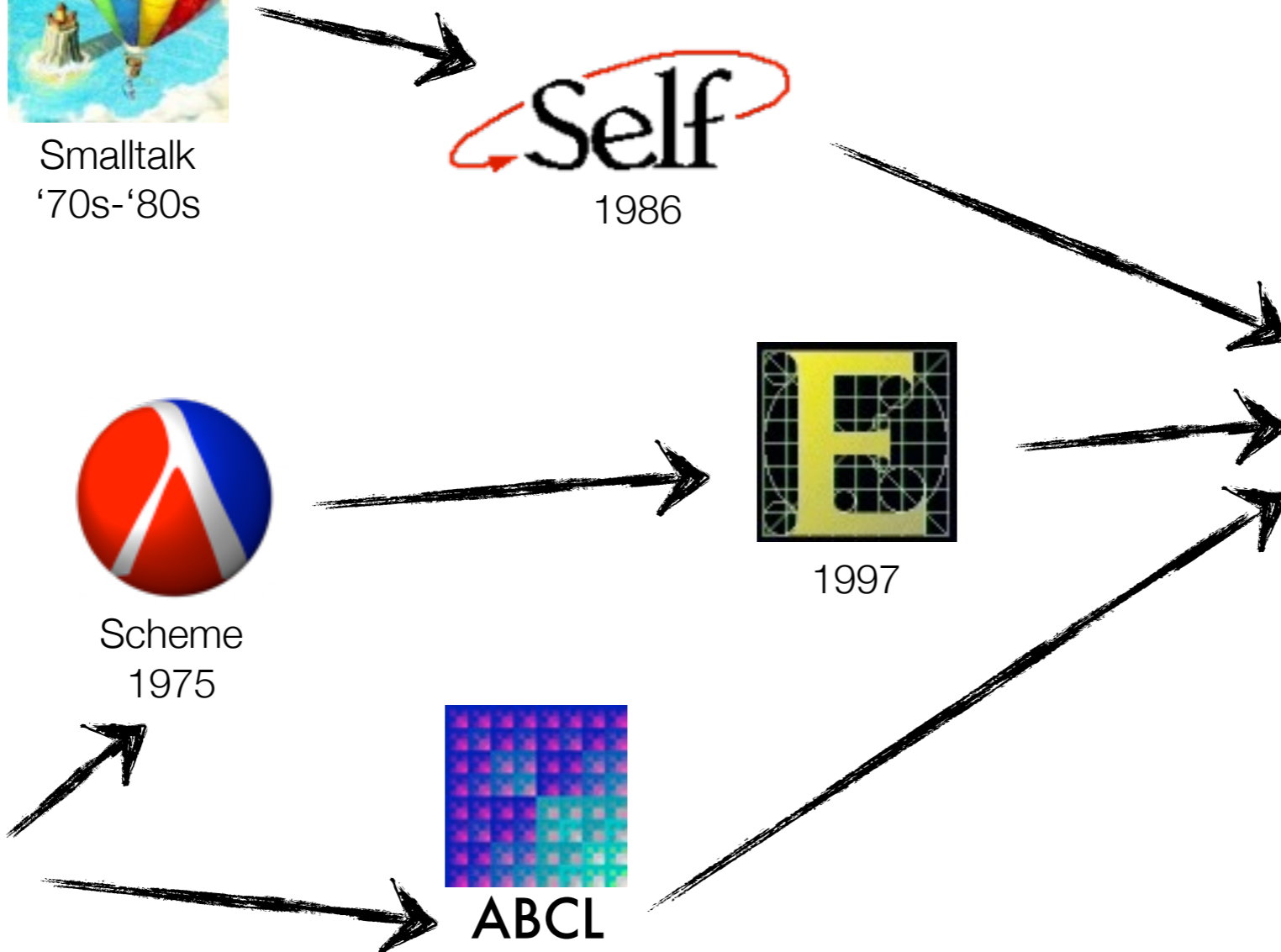


ABCL  
1986

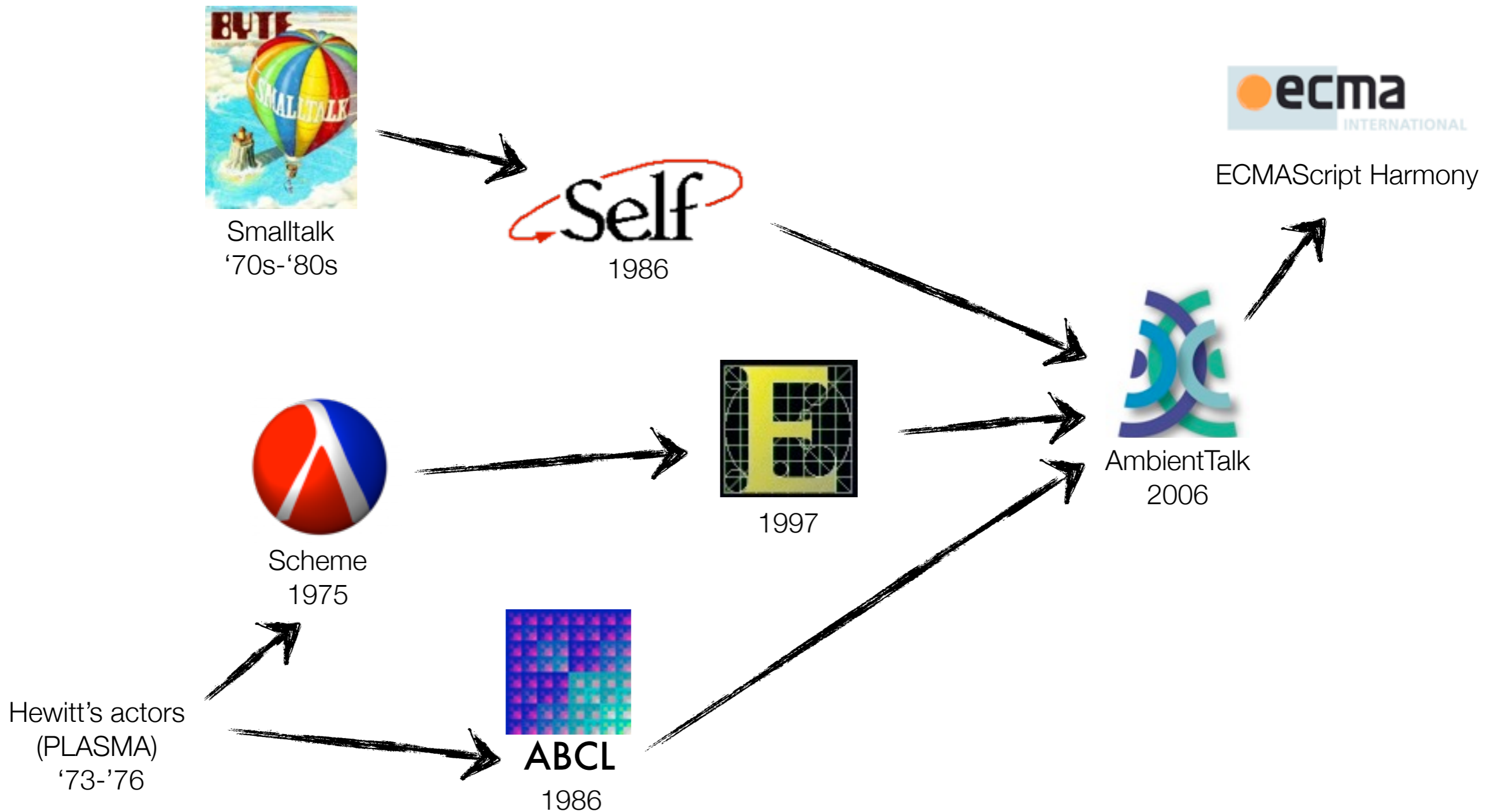


AmbientTalk  
2006

Hewitt's actors  
(PLASMA)  
'73-'76

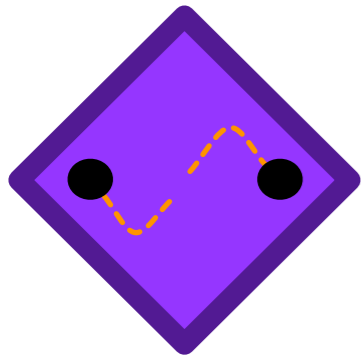


# Four Decades of Language Research





# How does AmbientTalk help?



## Volatile Connections



Asynchronous, buffered messaging  
send messages, even when disconnected



No blocking synchronization  
receive events, even when disconnected



Network failures  $\neq$  exceptions  
timeouts & leasing, whether connected or disconnected



## Zero Infrastructure

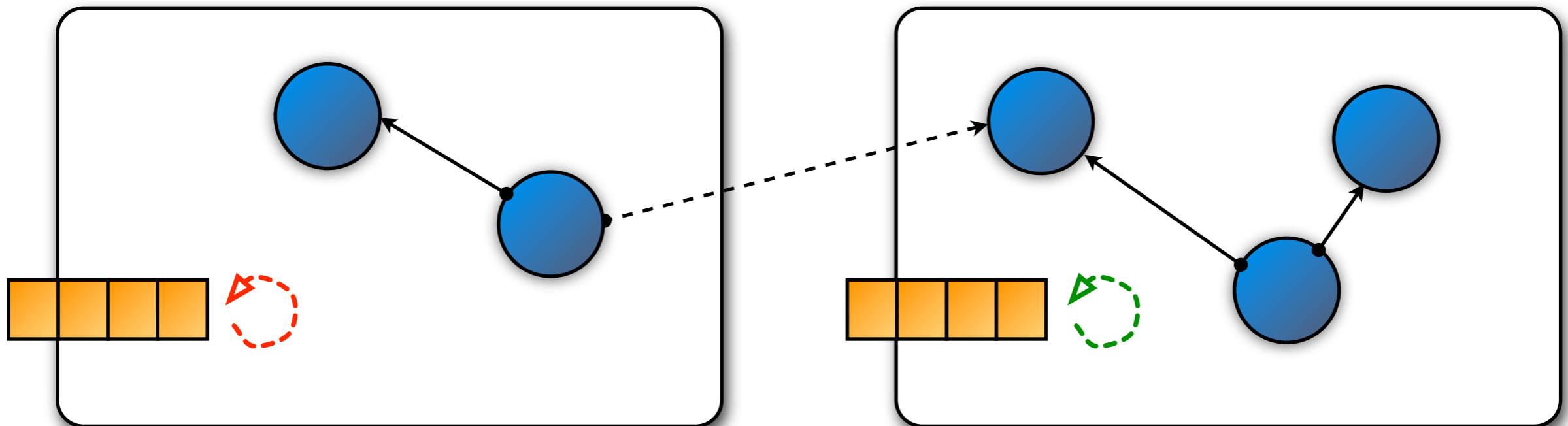


Peer-to-peer service discovery protocol  
decentralized, location-based



# Event Loop Concurrency

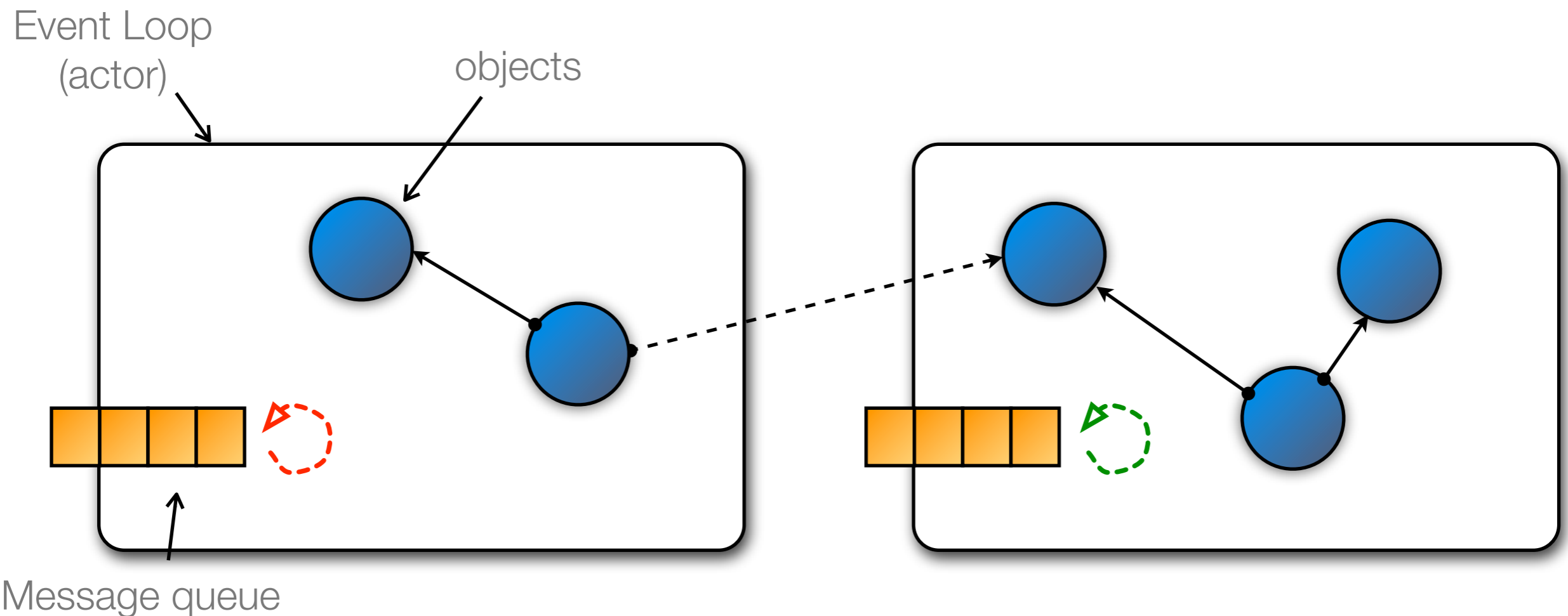
- AmbientTalk programs are **event loops**
- They **react** to events from the outside world
- Inter-event loop communication is **asynchronous**





# Event Loop Concurrency

- AmbientTalk programs are **event loops**
- They **react** to events from the outside world
- Inter-event loop communication is **asynchronous**



# Demo



# EchoServer

```
def service := object: {  
  def echo(text) {  
    system.println("Received: "+text);  
    text  
  }  
}  
  
deftype EchoService;  
  
def pub := export: service as: EchoService;
```

# EchoClient


```
deftype EchoService;  
  
def echoF := when: EchoService discovered: { |echoSvc|  
  system.println("Discovered an echo service");  
  echoSvc;  
} within: 2.minutes  
  
echoF<-echo("test1");  
  
def resultF := echoF<-echo("test2")@TwoWay;  
when: resultF becomes: { |value|  
  system.println("Reply: " + value);  
}  
  
echoF<-echo("test3");
```

# AmbientTalk = OO + Events

-  Generate and receive application requests  

```
obj<-msg(arg)  
def msg(param) { ... }
```
-  Follow-up on outstanding requests  

```
when: future becomes: { |result| ... }
```
-  React to services appearing and disappearing  

```
when: type discovered: { |ref| ... }
```
-  React to references disconnecting, reconnecting, expiring  

```
when: ref disconnected: { ... }  
when: ref reconnected: { ... }  
when: ref expired: { ... }
```

# Urbiflock

- P2P Geosocial networking framework
- Test deployment on Brussels public transport network



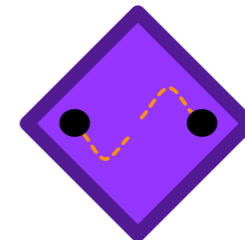
# Summary



ad hoc



Zero Infrastructure



Volatile Connections

# Summary

J2SE 1.5



J2ME CDC



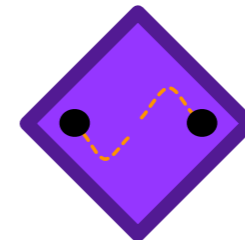
Android 1.6



ad hoc



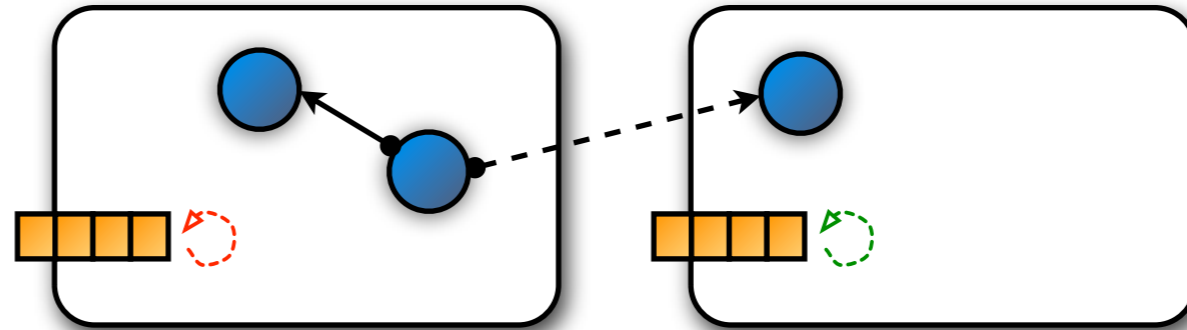
Zero Infrastructure



Volatile Connections



# Summary



**Decentralized  
Discovery**



**Asynchronous  
Communication**



**Non-blocking  
Synchronisation**



**Disconnections  
≠ Failures**

J2SE 1.5



J2ME CDC



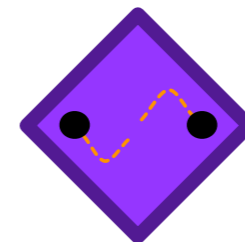
Android 1.6



ad hoc



Zero Infrastructure



Volatile Connections



[ambienttalk.googlecode.com](http://ambienttalk.googlecode.com)