KL1 言語とは

最も単純な並行論理型言語の一つ

1. 宣言的な記述が可能
   (a) 述語呼び出しによって, プロセスを動的に生成する
   (b) 共有変数を通信路として, 非同期プロセス間通信を行う
   (c) プロセス間で未定義変数を通信することによって, 通信路を動的に構成する (cf. π 計算の channel mobility)
2. 静的解析の技術が蓄積されている
   (a) 通信データやモバイルコードの検証・最適化を期待できる
3. 既存の実装
   (a) KLIC 処理系 = 翻訳系 + 実行時ライブラリ

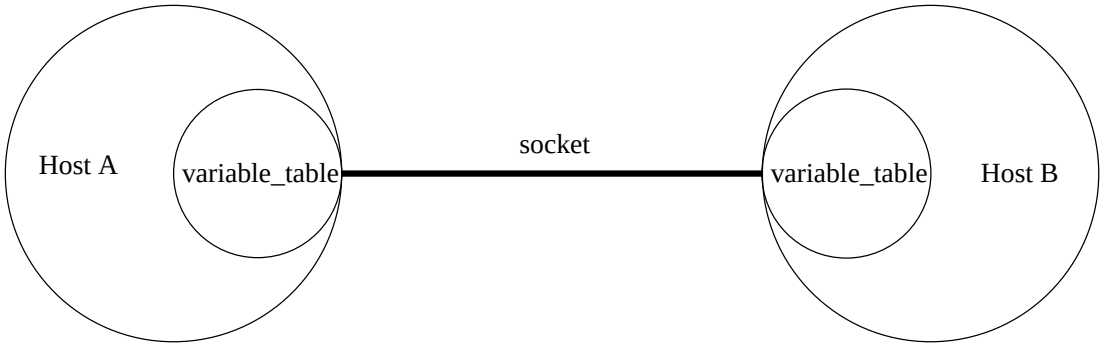
DKLIC とは

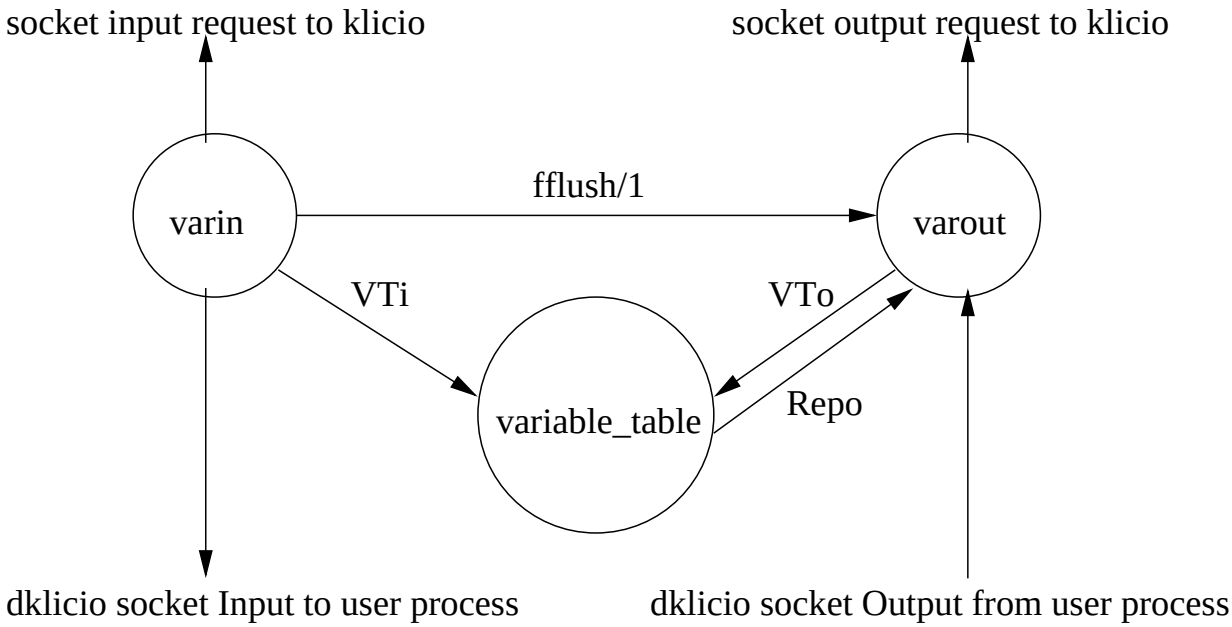
KL1 言語に基づく分散ミドルウェア（実行時ライブラリ）

1. 宣言型分散プログラミングを可能にした
2. how? KLIC 処理系を拡張して, 宣言型分散機能を付加した
3. what? 宣言型ソケット dklicio を実装した

宣言型ソケット dklicio とは

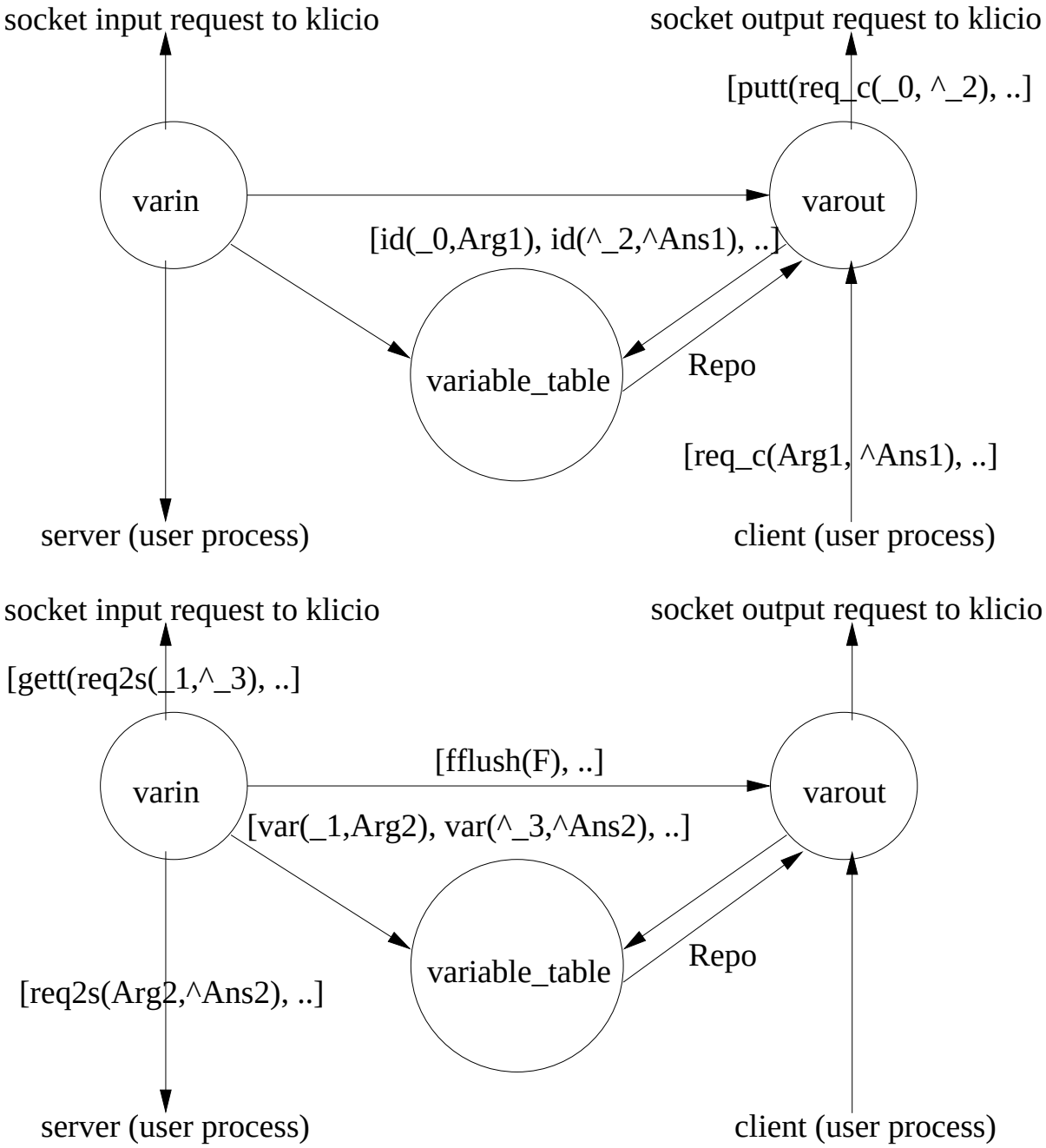
宣言型ソケット = 手続き型ソケット + 変数表





変数表とは

通訳系 : user 宣言型入出力 → socket 入出力手続き



分散論理変数 fusesusp とは

書き込み・直結・廃棄されたことを検出する分散論理変数オブジェクト

1. KL1 言語では, 未定義変数が直結・廃棄されたことを検出することができない
2. KLIC 処理系のジェネリックオブジェクト (cf. Java Native Interface) として, C 言語で実装した
3. モードを持たせることによって, dklicio は効率的な通信プロトコルを実装した

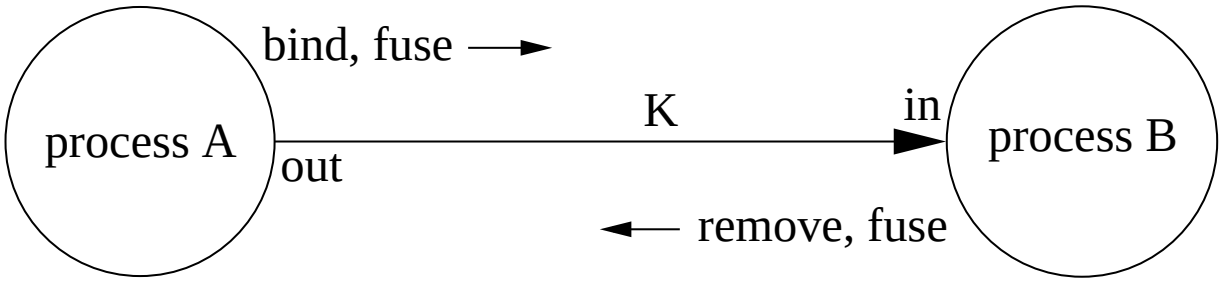
KLIC 処理系から呼び出される, fusesusp のメソッドは次の通り

1. コンストラクタ new(X, ID, Repo).
2. 書き込み bind(Term).
3. 直結 fuse(Y). (cf. π 計算の channel fusion)
4. 廃棄 remove.

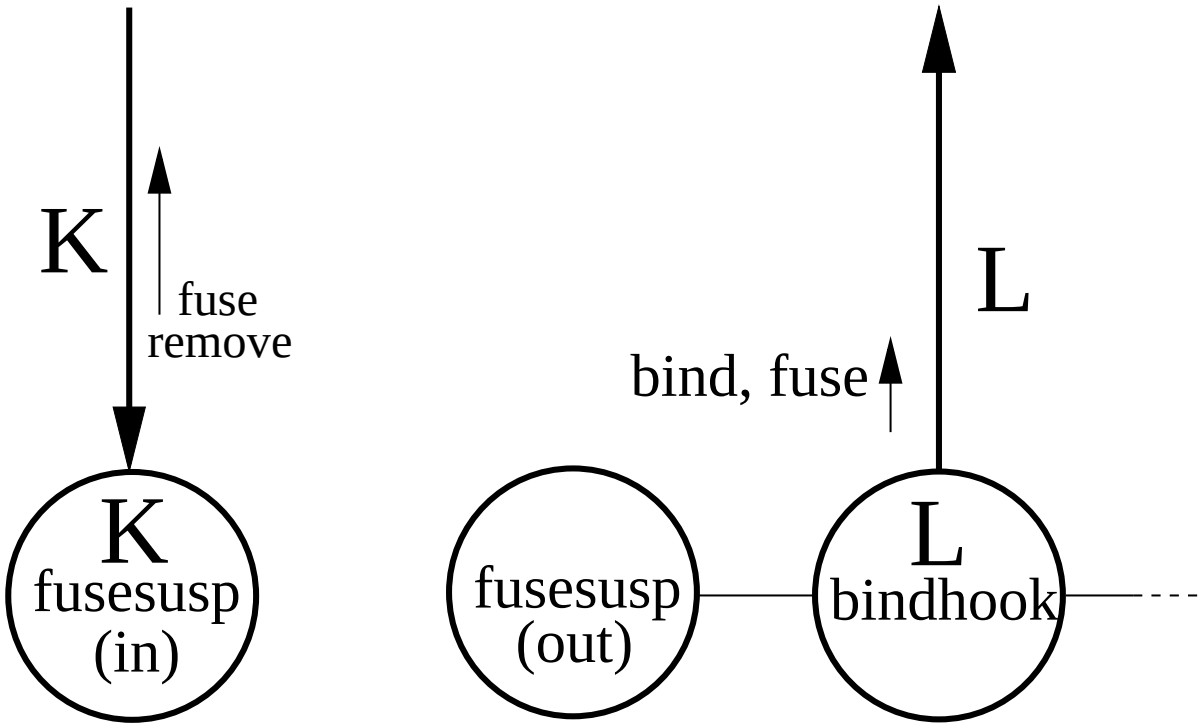
bind, fuse, remove を呼び出されると, fusesusp は変数表にその旨を報告する

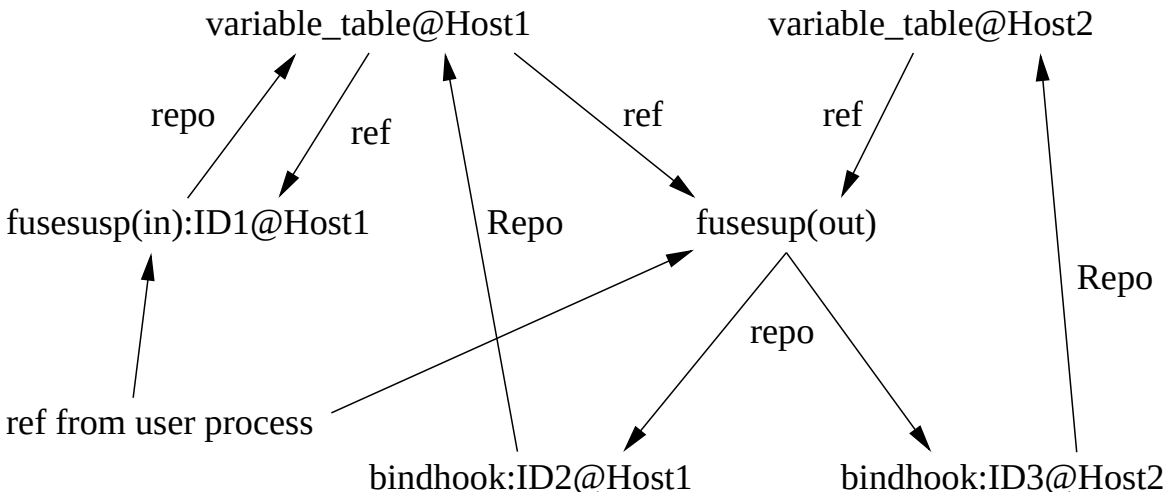
モードとは

1. 変数に対する書き込みの方向
2. KL1 言語における静的解析体系の一つ
3. 一つの通信路に対する書き手はちょうど一人
   (a) in モードの変数に書き込みは起きない
   (b) in モードの変数同士が直結されることはない
   (c) out モードの変数が廃棄されることはない

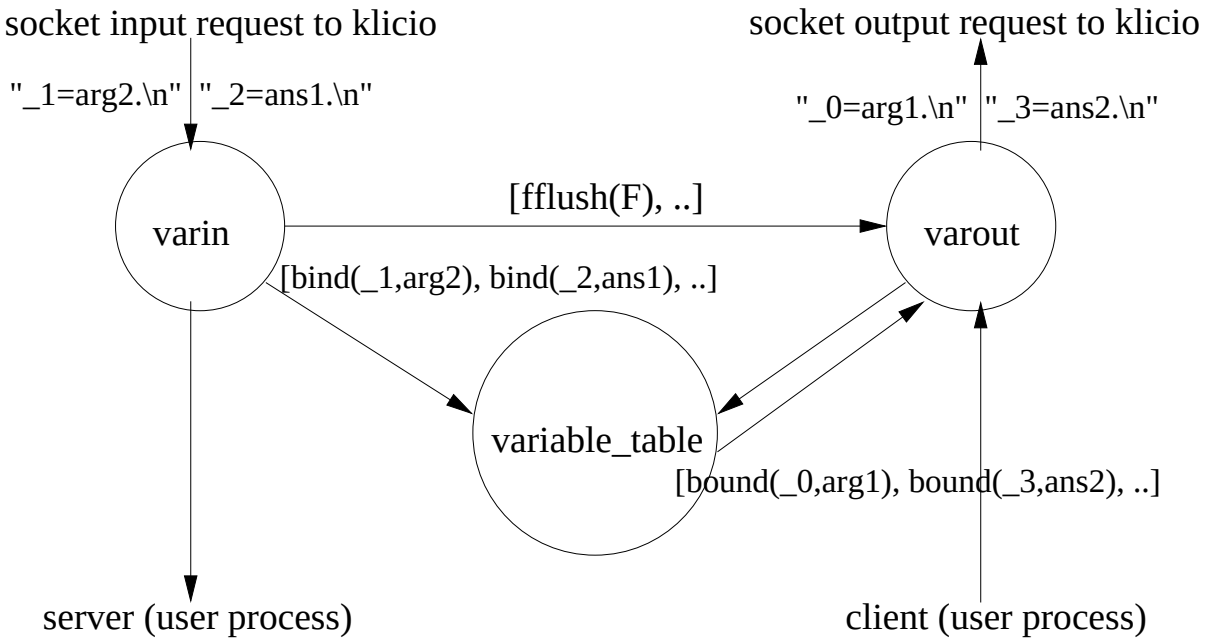


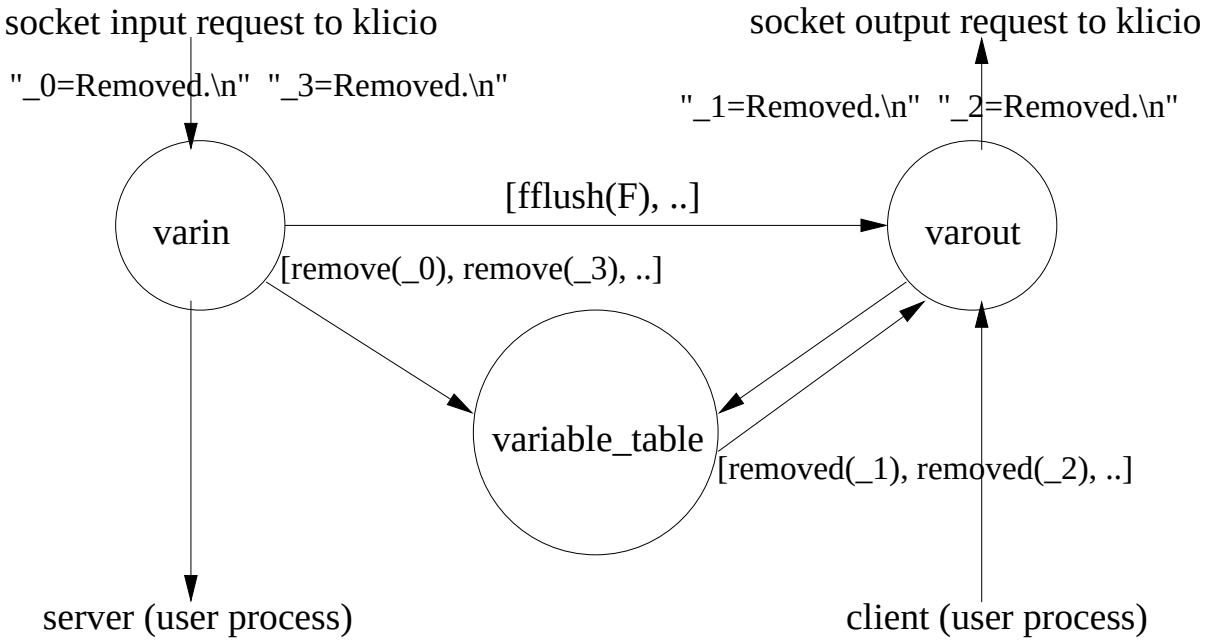
fusesusp in/out



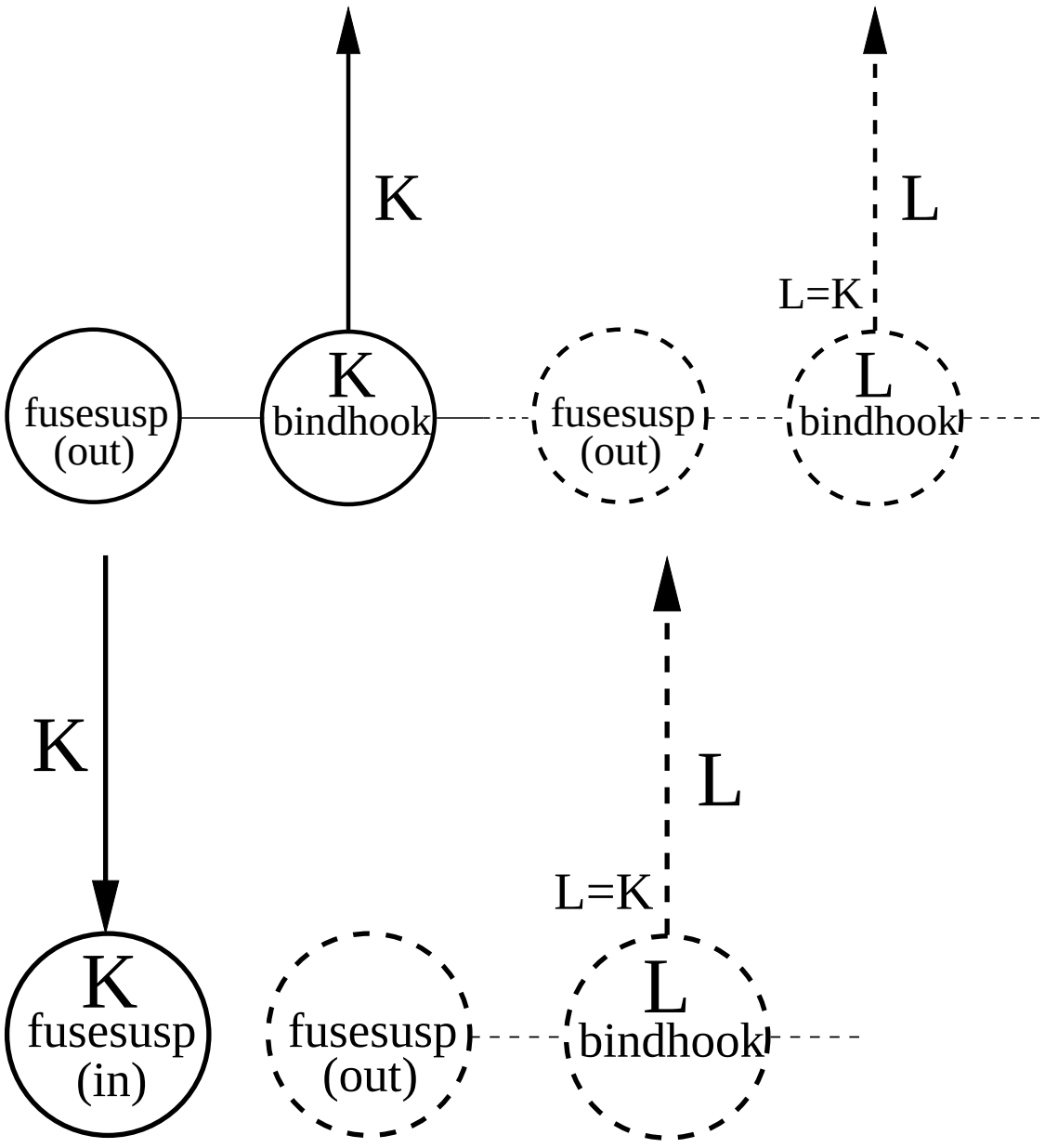


書き込みと廃棄

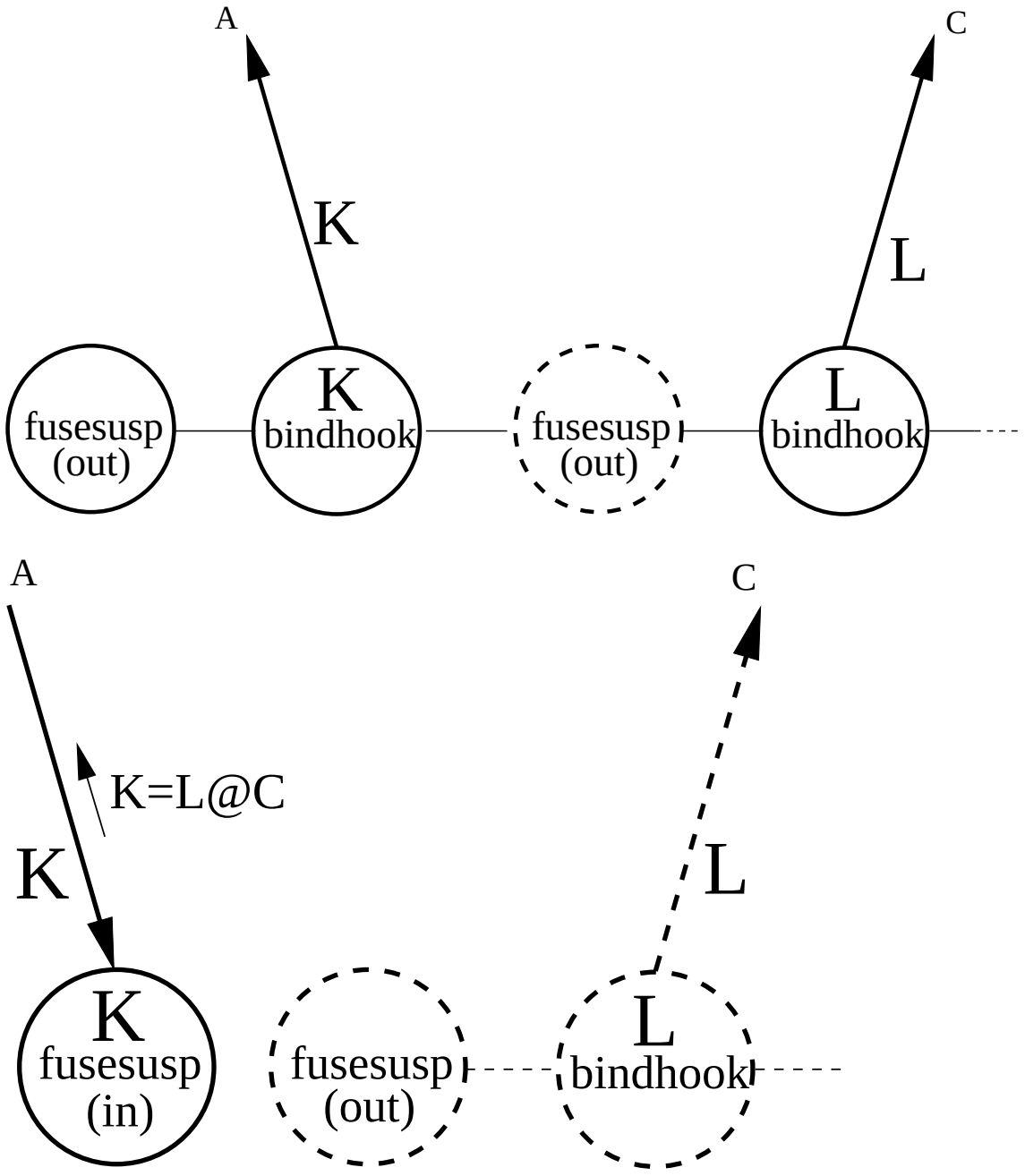

socket input request to klicio
"_1=arg2.\n" "_2=ans1.\n"
socket output request to klicio
"_0=arg1.\n" "_3=ans2.\n"
varin
[fflush(F), ..]
varout
[bind(_1,arg2), bind(_2,ans1), ..]
variable_table
[bound(_0,arg1), bound(_3,ans2), ..]
server (user process)
client (user process)

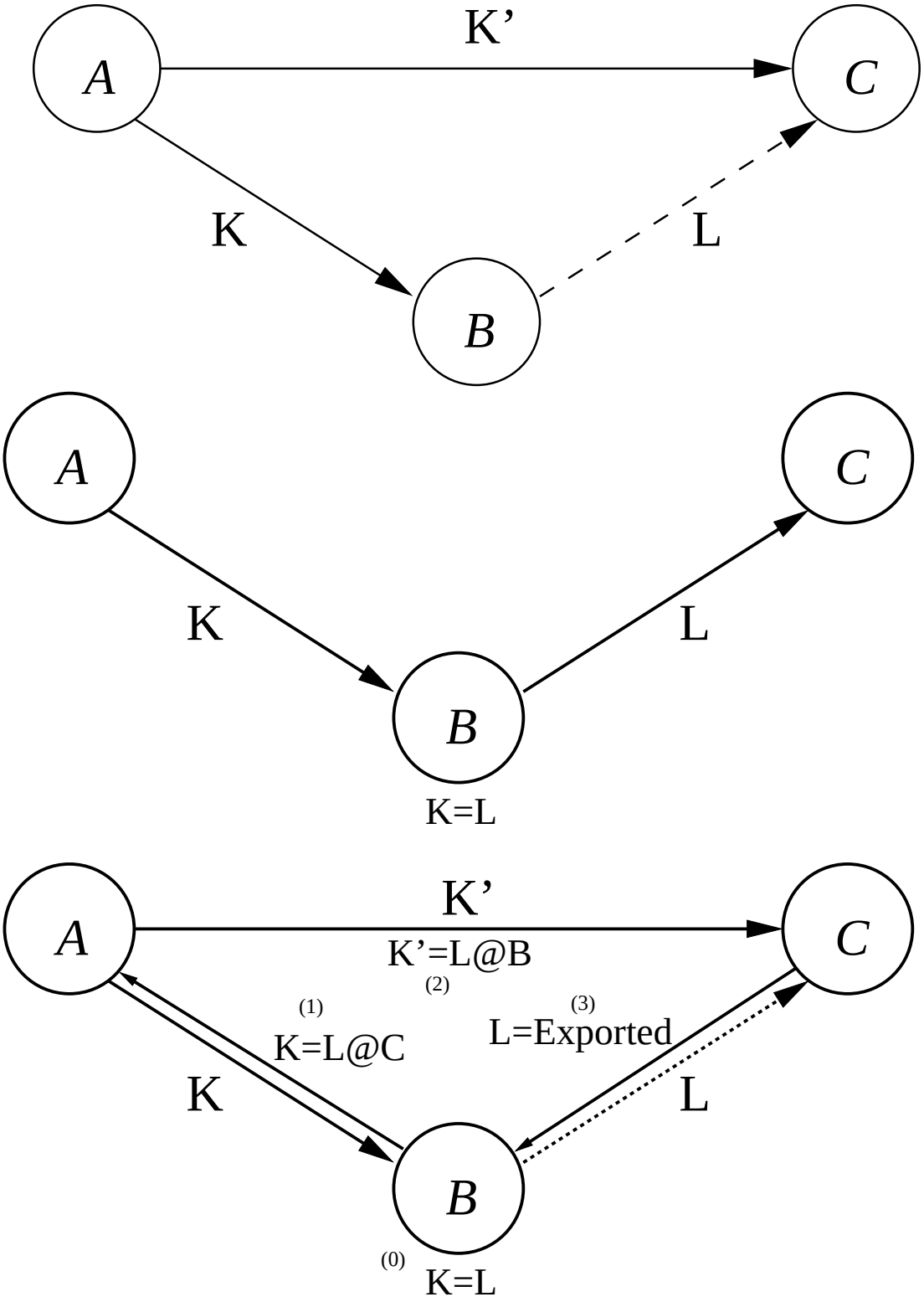

socket input request to klicio
"_0=Removed.\n" "_3=Removed.\n"
socket output request to klicio
"_1=Removed.\n" "_2=Removed.\n"
varin
[fflush(F), ..]
varout
[remove(_0), remove(_3), ..]
variable_table
[removed(_1), removed(_2), ..]
server (user process)
client (user process)

直結 — 二者通信（不要な通信の排除）



直結 — 三者通信（中継の排除）




A
K'
C
K
L
B
A
C
K
L
B
K=L
A
K'
C
K'=L@B
(2)
(1)
K=L@C
(3)
L=Exported
K
L
B
(0)
K=L

まとめ

1. 宣言型分散プログラミングを可能にした
2. 宣言型ソケット dklicio を実装した
3. 分散論理変数 fusesusp を実装した
   (a) 直結・廃棄されたことを検出することによって, 二重通信・中継を排除した
   (b) モードを導入することによって, 効率的な通信プロトコルを設計・実装した