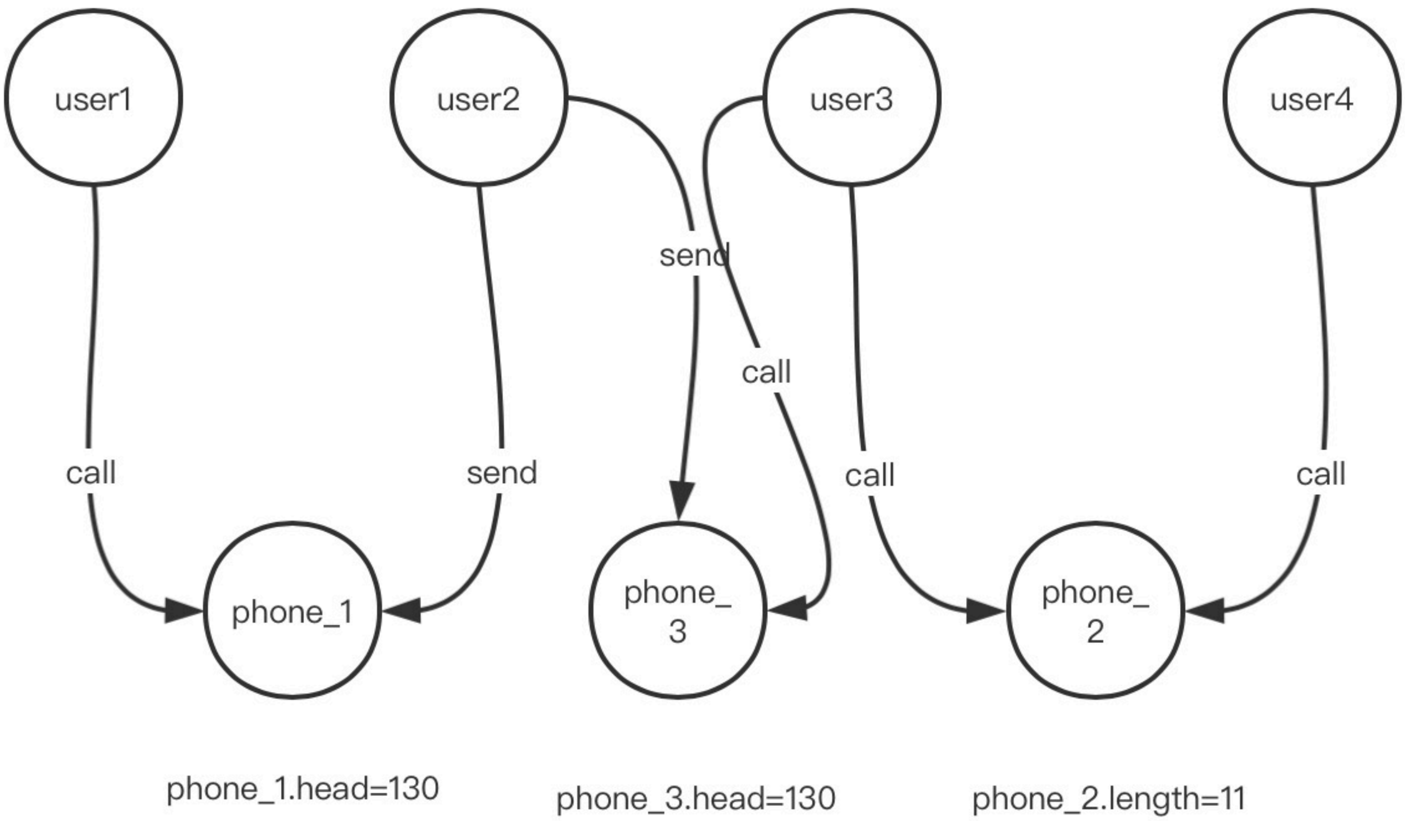
Graph With Attribute

表示形式

$$(V, E, A)$$

V :点集
E :边集
A :属性literal集合，每个属性literal的表示形式为$x.A = c$，其中x为节点编号，A为属性名，c为其值

例子



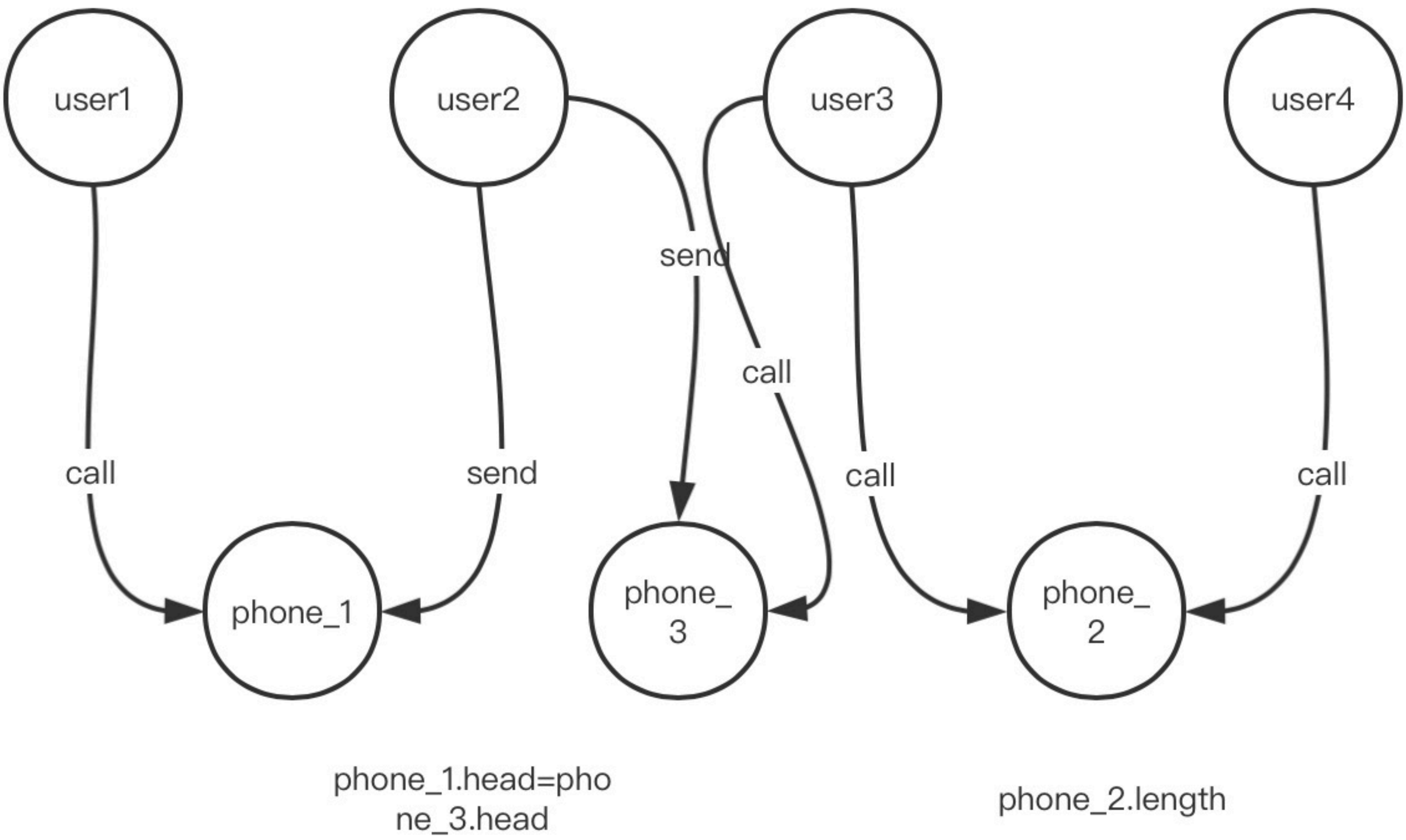
Graph Pattern With Attribute

表示形式

$$(V, E, A)$$

V :点集
E :边集
A :属性literal集合，每个属性literal的表示形式为$x.A=c, x.A=y.B, x.A$

例子



Attribute Expansion Graph

表示形式

$$(V, E, V_A, V_V, V_C, E_A, E_V, E_C)$$

V :原图中的点集
E :原图中的边集
V_A :Attribute点
V_V :Value点
V_C :Constant点
E_A :$E_A \in V * V_A$，其中(u, v)表示一条从u到v的有向边
E_V :$E_V \in V_A * V_V$，其中(u, v)表示一条从u到v的有向边

E_C :$E_C \in V_V * V_C$，其中(u,v)表示一条从u到v的有向边

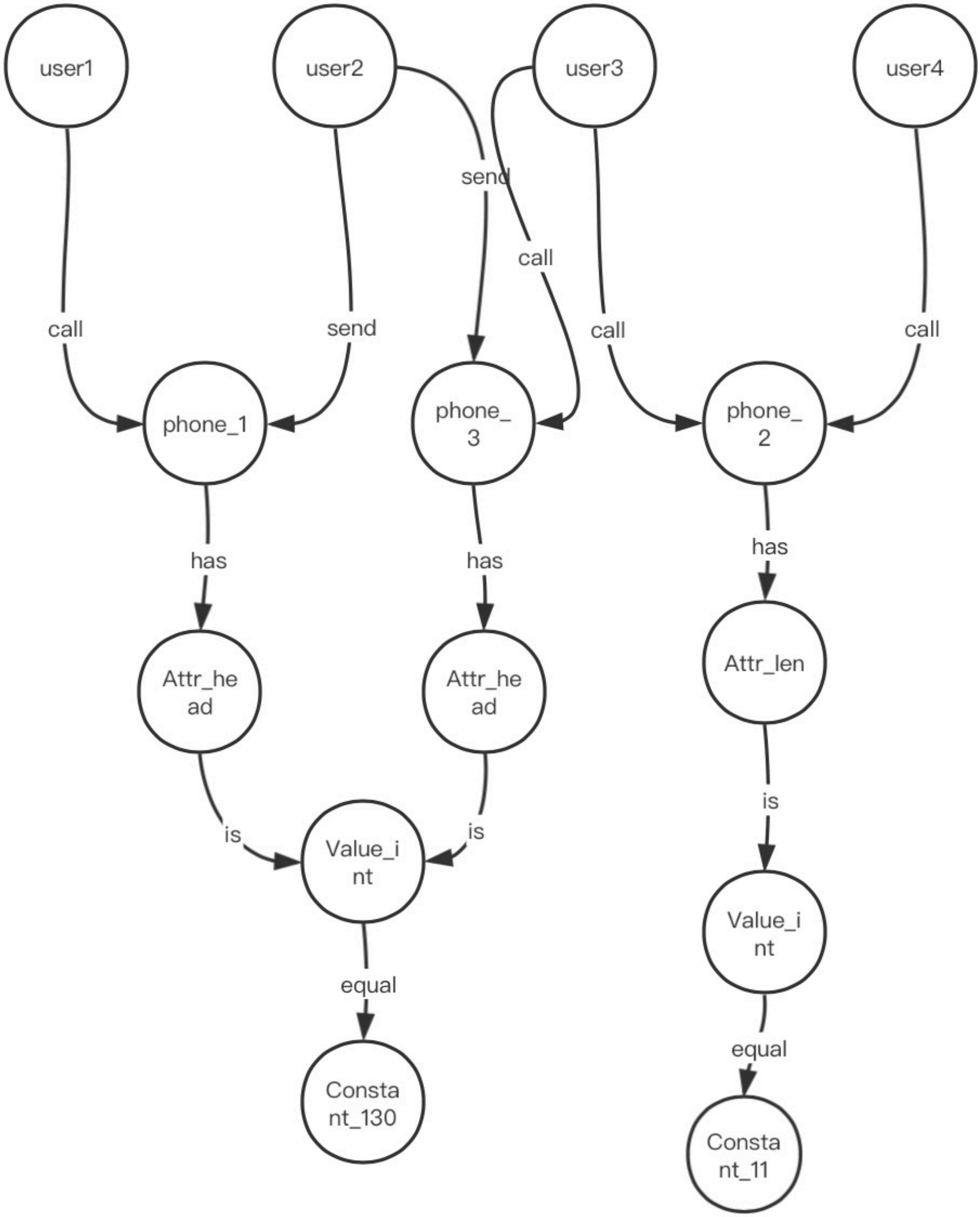
限制条件

1.若$(u,v) \in E_C$,则不存在$u' \neq u$使得$(u',v) \in E_C$
2.若$(u,v) \in E_C$,则不存在$v' \neq v$使得$(u,v') \in E_C$
3.若$(u,v) \in E_V$,则不存在$v' \neq v$使得$(u,v') \in E_C$
4.若$(u,v) \in E_A$,则不存在$u' \neq u$使得$(u',v) \in E_C$
5.对于$v \in V_A$,一定有$indegree(v) = 1, outdegree(v) = 1$
6.对于$v \in V_V$,一定有$indegree(v) \geq 1, outdegree(v) = 1$
7.对于$v \in V_C$,一定有$indegree(v) = 1, outdegree(v) = 0$

说明

1.value点与constant点一一对应，有value点一定就要有对应的constant点
2.一个attribute点只能连向一个value点，一个value点可与多个Attribute点一一对应
3.一个实体点可与多个Attribute点对应，一个Attribute点只能对应一个实体点

例子



Graph With Attribute转化为Attribute Expansion Graph

对于Graph With Attribute.A中的每个literal，进行如下操作：

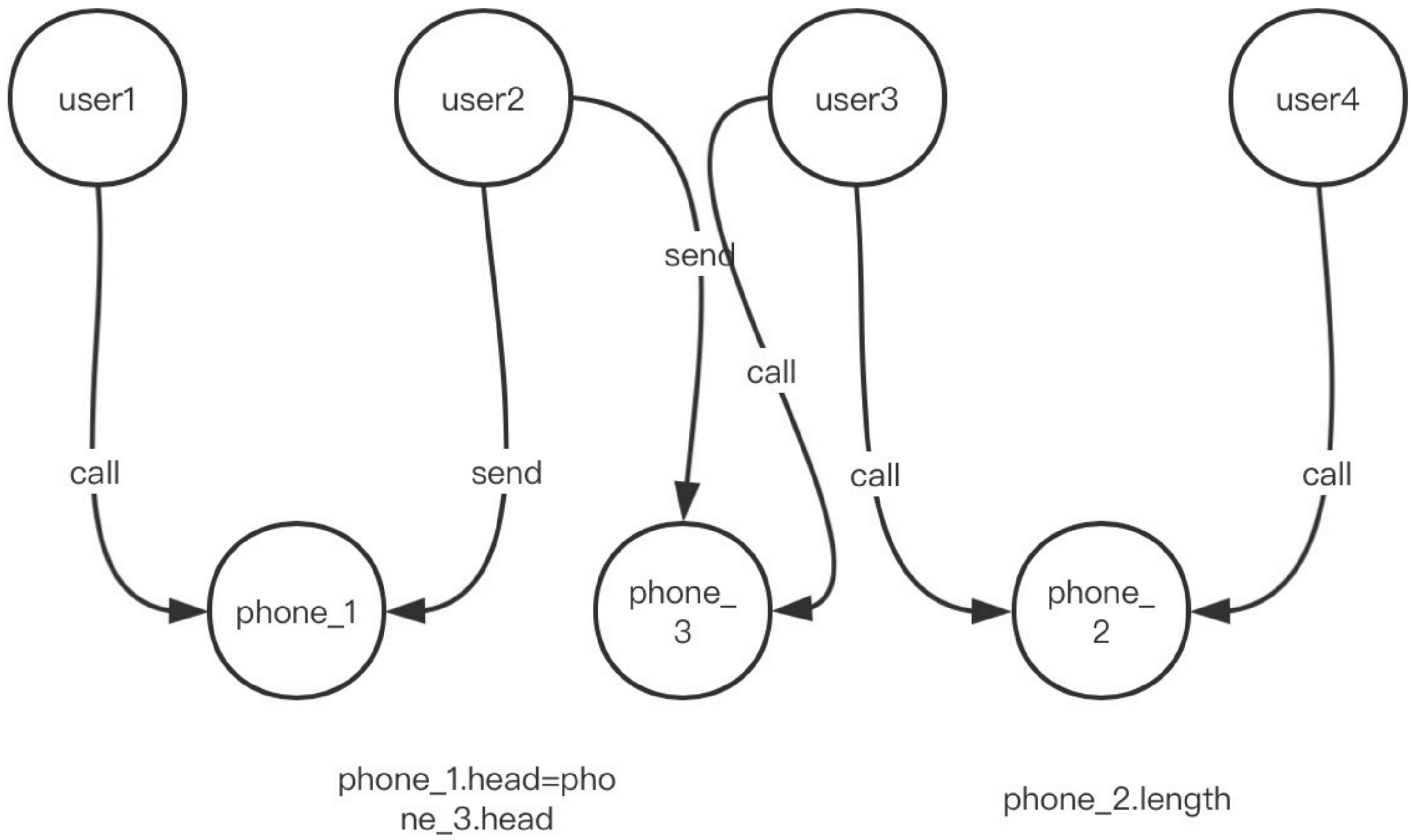
1.在x节点后添加一个Attribute节点，表示其属性A，并将x节点与Attribute节点脸边

2.若图中存在表示值为c的Value节点，则将该Attribute节点与其连边，否则在该Attribute节点后添

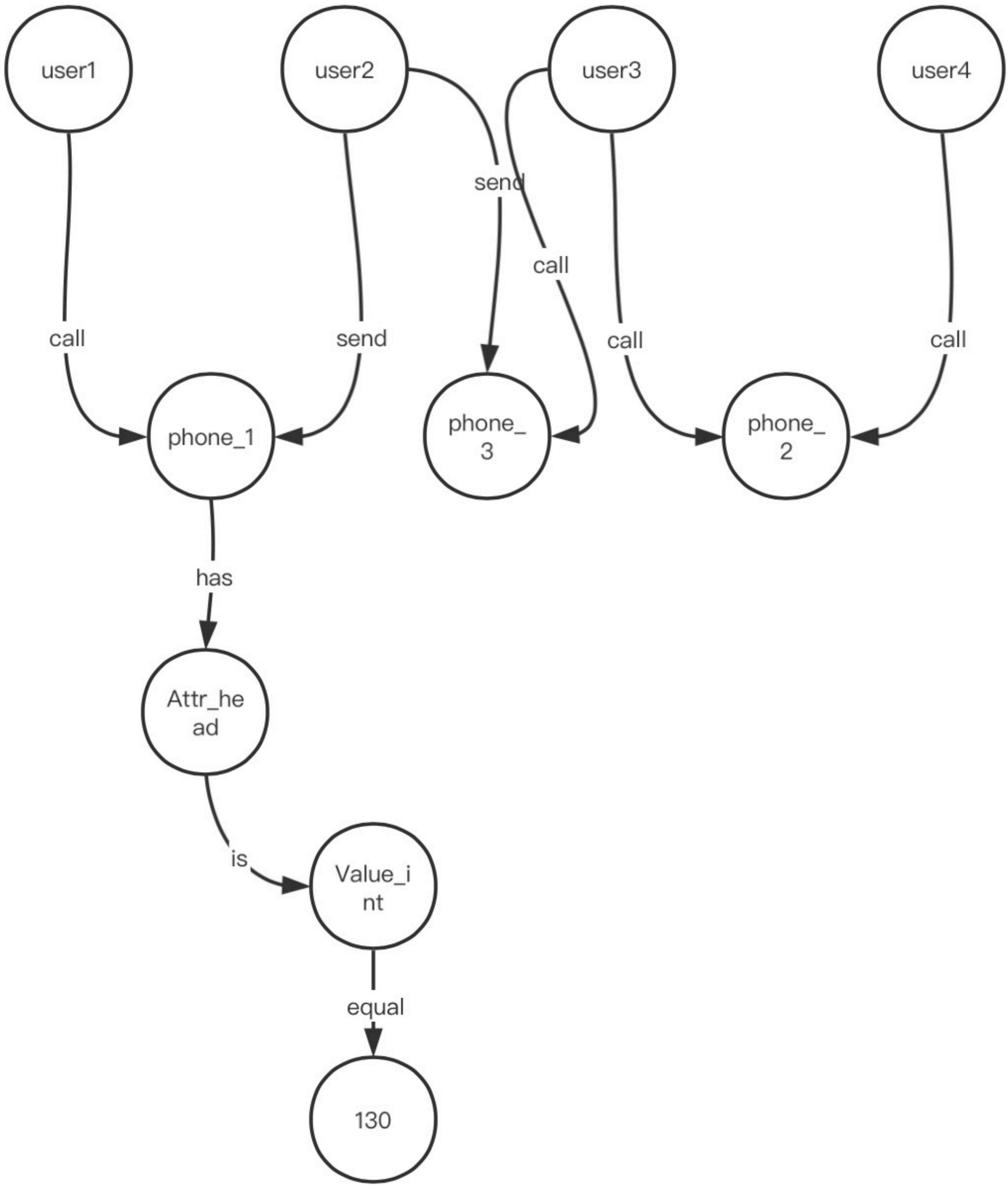
加一个Value节点并添加从Attribute节点与Value节点的有向边
3.若操作2中的Value节点为新加入节点，则在该Value节点后添加一个label为c的constant节点，表示其值为constant，并将该Value节点与Constant节点相连

例子

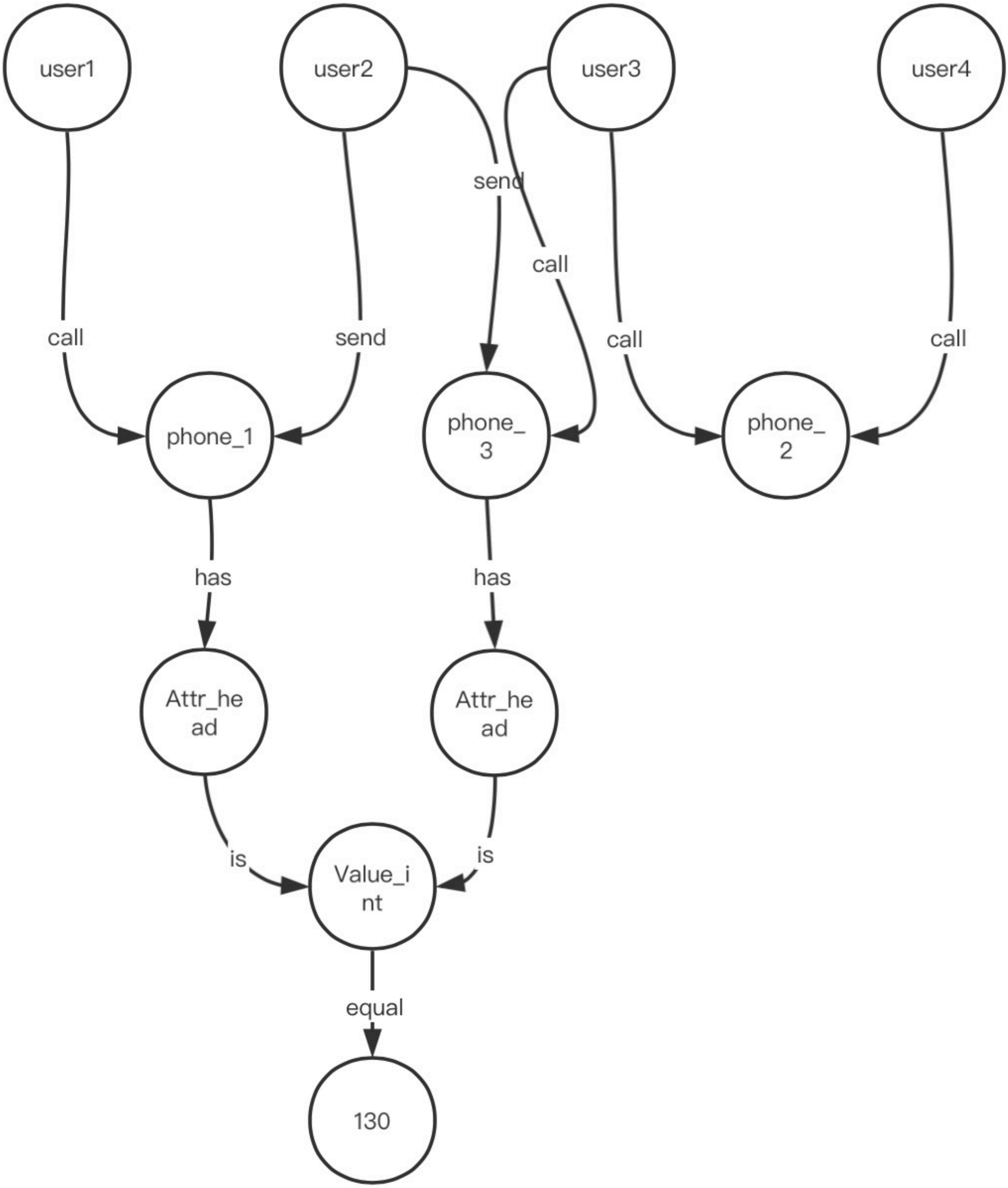
对于



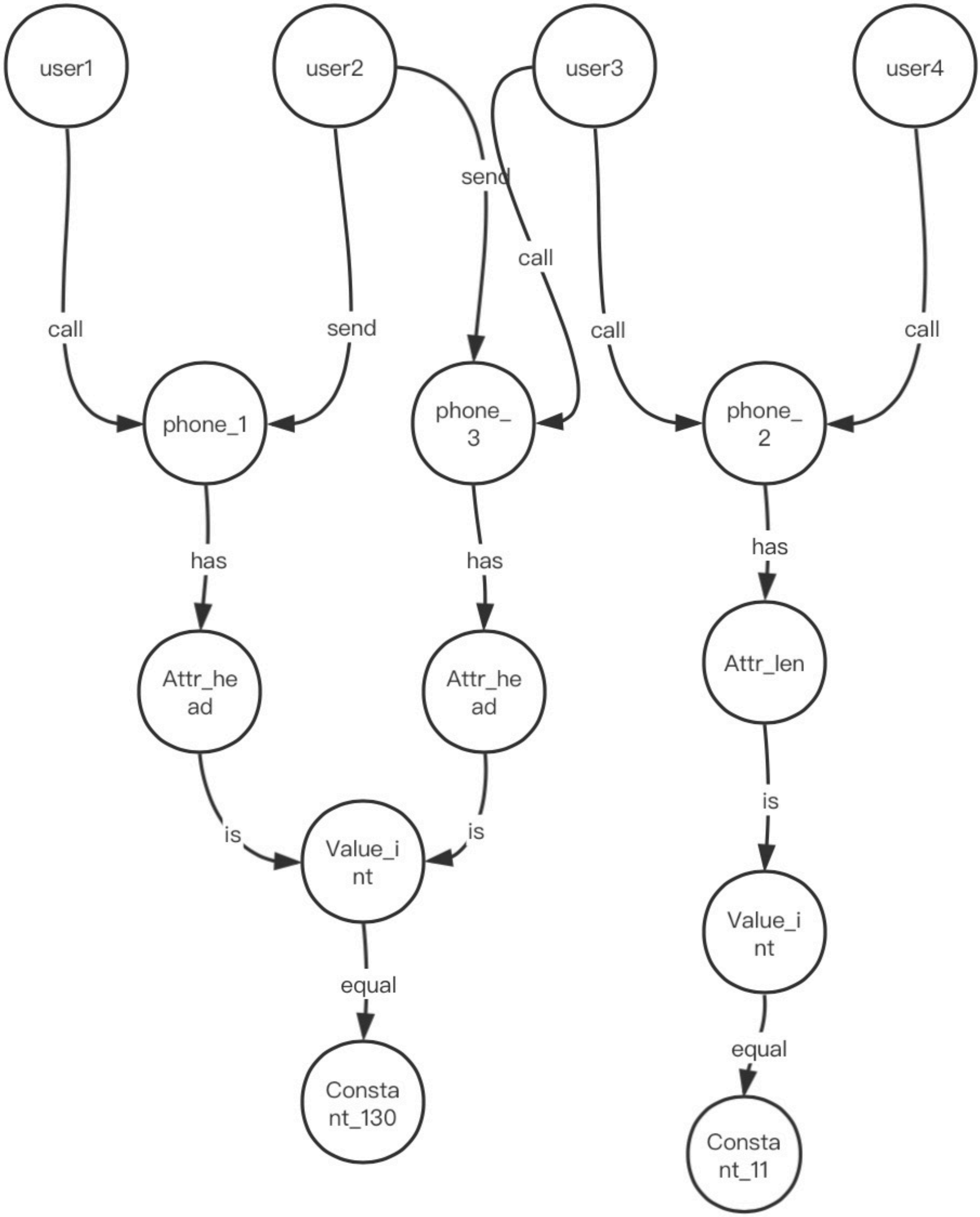
首先，展开literal $phone_1.head = 130$，得到如下结果



接着，展开literal $phone_3.head = 130$，由于Value为130的Value节点已经存在，不单独加点，得到如下结果



最后，展开literal $phone_2.length = 11$,得到如下结果



Attribute Expansion Graph Pattern

表示形式

$$(V, E, V_A, V_V, V_C, E_A, E_V, E_C)$$

V :原图中的点集
E :原图中的边集
V_A :Attribute点
V_V :Value点
V_C :Constant点
E_A :$E_A \in V * V_A$，其中(u, v)表示一条从u到v的有向边
E_V :$E_V \in V_A * V_V$，其中(u, v)表示一条从u到v的有向边
E_C :$E_C \in V_V * V_C$，其中(u, v)表示一条从u到v的有向边

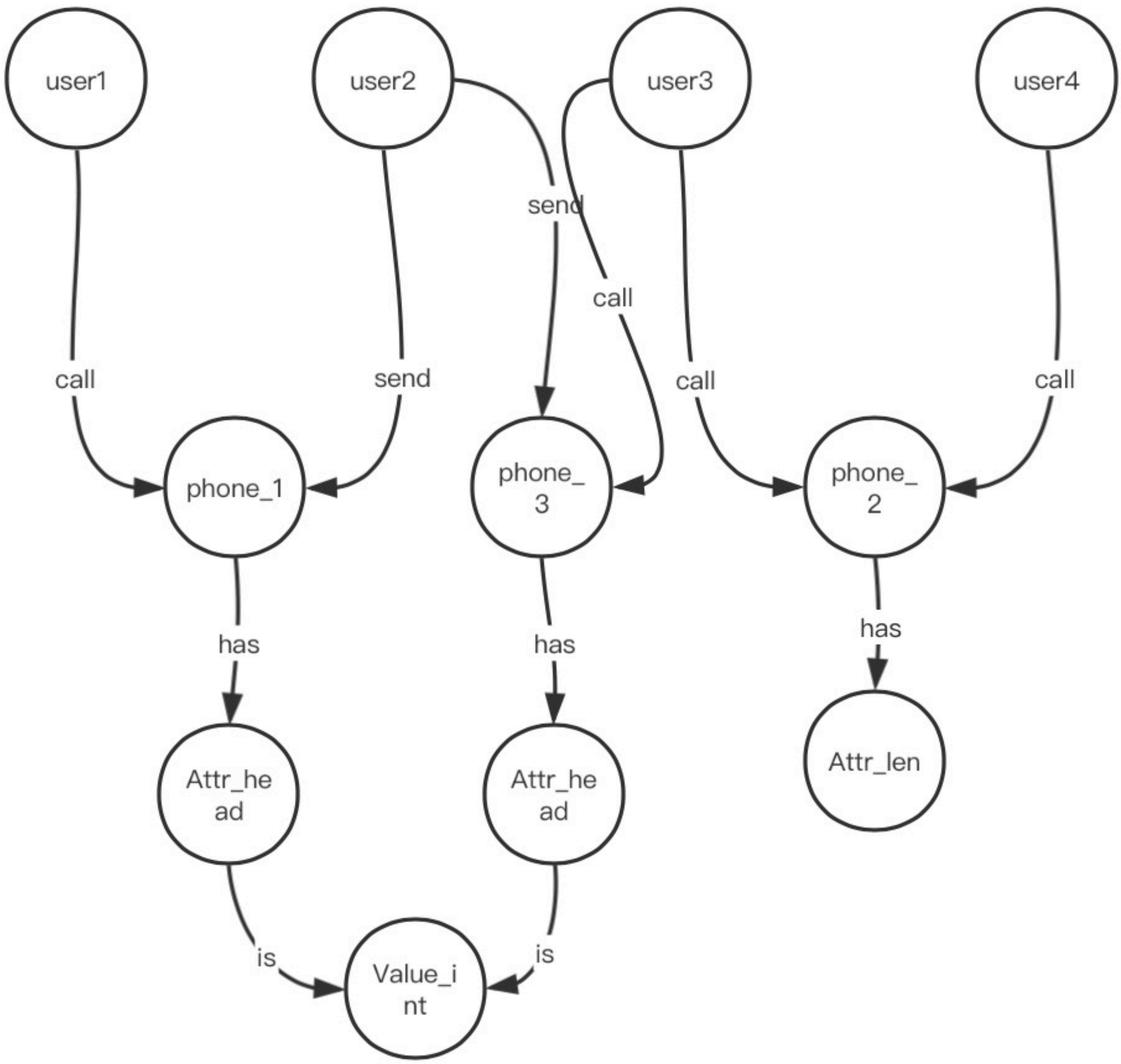
限制条件

1.若$(u, v) \in E_C$,则不存在$u' \neq u$使得$(u', v) \in E_C$
2.若$(u, v) \in E_C$,则不存在$v' \neq v$使得$(u, v') \in E_C$
3.若$(u, v) \in E_V$,则不存在$v' \neq v$使得$(u, v') \in E_C$
4.若$(u, v) \in E_A$,则不存在$u' \neq u$使得$(u', v) \in E_C$
5.对于$v \in V_A$,一定有$indegree(v) = 1, outdegree(v) = 1$
6.对于$v \in V_V$,一定有$indegree(v) \geq 1, outdegree(v) \leq 1, degree(v) \geq 2$
7.对于$v \in V_C$,一定有$indegree(v) = 1, outdegree(v) = 0$

说明

1.value点与constant点一一对应,若value点没有对应的constant点，需要有多个attribute点对应该value点
2.一个attribute点只能连向一个value点，一个value点可与多个Attribute点一一对应
3.一个实体点可与多个Attribute点对应，一个Attribute点只能对应一个实体点

例子



性质

每一个Attribute Expansion Graph Pattern均可对应一个Graph Pattern+Literal Set，且每一个Graph Pattern+Literal Set的结合体均可表示为一个Attribute Expansion Graph Pattern。其中LiteralSet支持的语义有：

1.x.A

2.x.A=c

3.x.A=y.B

证明

先证明：每一个Attribute Expansion Graph Pattern均可对应一个Graph Pattern+Literal Set。
若$V_A = V_V = V_C = \emptyset$，则对应Graph Pattern+$\emptyset$
若$V_A \neq \emptyset$, $V_V = V_C = \emptyset$，则对应Graph Pattern+$x.A$,其中x为Attribute点对应的实体点
若$V_A \neq \emptyset$, $V_V \neq \emptyset$, $V_C = \emptyset$，此时的value点一定有多个Attribute点对应，此时可以对应的结构有：Graph Pattern+$x.A = y.B$,Graph Pattern+$x.A, x.A = y.B$
若$V_A \neq \emptyset$, $V_V \neq \emptyset$, $V_C \neq \emptyset$，此时的value点要么有多个Attribute点对应，要么有一个constant点对应。此时可以对应的结构有：Graph Pattern+$x.A = y.B, x.A = c$,Graph Pattern+$x.A, x.A = y.B, x.A = c$,Graph Pattern+$x.A, x.A = c$,Graph Pattern+$x.A = c$
此时，Graph Pattern+Literal Set的全部情况均对应上

再证明：对于任意的一种raph Pattern+Literal Set,都可以转化为一个Attribute Expansion Graph Pattern
对于x.A，在x节点处拓展一个Attribute节点即可
对于x.A=y.B，在x节点拓展一个Attribute节点A,在y节点处拓展一个Attribute B，将A,B节点与同一个value节点相连
对于x.A=c,在x节点拓展一个Attribute节点A，在A节点拓展一个value节点V,在V节点拓展一个constant节点C即可

挖掘算法

基于传统方法的挖掘算法（一条一条加边）做一定变动，即将加边操作变为实体拓展或属性拓展，挖掘的是在一定的操作步数内能够得到的Attribute Expansion Graph Pattern
实体拓展：正常加边
属性拓展：分为以下几种情况
1.在某个实体节点拓展一个A节点
2.选择>=2个Attribute节点，新建一个Value节点且让选择的Attribute节点与该节点连边
3.选择一个Value节点，拓展一个constant节点