9 Role Assignments

Jürgen Lerner

Classification is the key to understand large and complex systems that are made up of many individual parts. For example in the study of food webs (networks that consist of living organisms and predator-prey relationships, flow of protein, etc.) it is, even for moderately small ecosystems, impossible to understand the relationship between each pair of individual organisms. Nevertheless, we can understand the system – to a certain extent – by classifying individuals and describing relationships on the class level. Classification in networks aims to describe regular patterns of interaction and to highlight essential structure, which remains stable over long periods of time.

In this chapter we formalize the classification of vertices in a graph, such that vertices in the same class can be considered to occupy the same *position*, or play the same *role* in the network. This idea of network position or role, see e. g., Nadel [436], has been formalized first by Lorrain and White [394] by a special type of vertex partition. They proposed that vertices play the same role if they have identical neighborhoods. Subsequent work like Sailer [501] and White and Reitz [579] generalized this early definition, weakening it sufficiently to make it more appropriate for modeling social roles. All these definitions have in common that vertices which are claimed to play the same role must have something in common w. r. t. the relations they have with other vertices, i. e., a generic problem definition for this chapter can be given by

given a graph $G = (V, E)$,
find a partition of V that is *compatible* with E.

The generic part here is the term 'compatible with E'. In this chapter, we present definitions for such compatibility requirements, and properties of the resulting classes of vertex-partitions.

Outline of this chapter. The remainder of this section treats preliminary notation. In Sections 9.1 through 9.3, different types of role assignments are introduced and investigated. In Section 9.4 definitions are adapted to graphs with multiple relations (see Definition 9.4.1) and in Section 9.5 composition of relations is introduced and its relationship to role assignments is investigated.

Sections 9.1 through 9.3 follow loosely a common pattern: After defining a compatibility requirement, some elementary properties of the so-defined set of role assignment are mentioned. Then, we investigate a partial ordering on this set, present an algorithm for computing specific elements, and treat the complexity of some decision problems. We provide a short conclusion for each type of vertex partition, where we dwell on the applicability for defining role assignments in empirical networks.

The most complete investigation is for regular equivalences in Section 9.2. Although there is some scepticism as to whether regular equivalences are a good formalization of role assignments in real social networks, we have chosen to treat them prominently in this chapter, since their investigation is exemplary for the investigation of types of role assignments. The results for regular equivalences are often translatable to other types of equivalences, often becoming easier or even trivial. We emphasize this generality when appropriate.

Graph model of this chapter. In this chapter, *graph* usually means directed graph, possibly with loops. Except for Sections 9.2.4 and 9.2.5, where graph means undirected graph, Section 9.3.1, where results are for undirected multigraphs, and Sections 9.4 and 9.5, where we consider graphs with multiple relations (see Definition 9.4.1).

9.0.1 Preliminaries

In the following, we will often switch between vertex partitions, equivalence relations on the vertex set, or role assignments, since, depending on the context, some point of view will be more intuitive than the other. Here we establish that these are just three different formulations for the same underlying concept.

Let V be a set. An *equivalence relation* $\sim$ is a binary relation on V that is *reflexive*, *symmetric*, and *transitive*, i. e., $v \sim v$, $u \sim v$ implies $v \sim u$, and $u \sim v \wedge v \sim w$ implies $u \sim w$, for all $u, v, w \in V$. If $v \in V$ then $[v] := \{u\,;\ u \sim v\}$ is its *equivalence class.*

A *partition* $\mathcal{P} = \{C_1, \dots, C_k\}$ of V is a set of non-empty, disjoint subsets $C_i \subseteq V$, called *classes* or *blocks*, such that $V = \bigcup_{i=1}^{k} C_i$. That is, each vertex $v \in V$ is in exactly one class.

If $\sim$ is an equivalence relation on V, then the set of its equivalence classes is a partition of V. Conversely, a partition $\mathcal{P}$ induces an equivalence relation by defining that two vertices are equivalent iff they belong to the same class in $\mathcal{P}$. These two mappings are mutually inverse.

Definition 9.0.1. *A* role assignment *for V is a surjective mapping $r\colon V \to W$ onto some set W of* roles.

The requirement *surjective* is no big loss of generality since we can always restrict a mapping to its image set. One could also think of role assignments as vertex-colorings, but note that we do not require that adjacent vertices must have different colors. We use the terms role and position synonymously.

A role assignment defines a partition of V by taking the inverse-images $r^{-1}(w) := \{v \in V\,;\ r(v) = w\}$, $w \in W$ as classes. Conversely an equivalence relation induces a role assignment for V by the class mapping $v \mapsto [v]$. These two mappings are mutually inverse, up to isomorphism of the set of roles.

We summarize this in the following remark.

Remark 9.0.2. For each partition there is a unique associated equivalence relation and a unique associated role assignment and the same holds for all other combinations.

For the remainder of this chapter, definitions for vertex partitions translate to associated equivalence relations and role assignments.

9.0.2 Role Graph

The image set of a role assignment can be supplied naturally with a graph structure. We define that roles are adjacent if there are adjacent vertices playing these roles:

Definition 9.0.3. *Let $G = (V, E)$ be a graph and $r\colon V \to W$ a role assignment. The* role graph *$R = (W, F)$ is the graph with vertex set W (the set of roles) and edge set $F \subseteq W \times W$ defined by*

$$F := \{(r(u), r(v))\,;\ \exists u, v \in V \text{ such that } (u, v) \in E\} \ .$$

R is also called quotient *of G over r.*

The role graph R models roles and their relations. It can also be seen as a smaller model for the original graph G. Thus, a role assignment can be seen as some form of network compression. Necessarily, some information will get lost by such a compression. The goal of role analysis is to find role assignments such that the resulting role graph displays essential structural network properties, i. e., that not too much information will get lost.

Thus we have two different motivations for finding good role assignments. First to know which individuals (vertices) are 'similar'. Second to reduce network complexity: If a network is very large or irregular, we can't capture its structure on the individual (vertex) level but perhaps on an aggregated (role) level. The hope is that the role graph highlights essential and more persistent network structure. While individuals come and go, and behave rather irregularly, roles are expected to remain stable (at least for a longer period of time) and to display a more regular pattern of interaction.

9.1 Structural Equivalence

As mentioned in the introduction, the goal of role analysis is to find meaningful vertex partitions, where 'meaningful' is up to some notion of compatibility with the edges of the graph. In this section the most simple, but also most restrictive requirement of compatibility is defined and investigated. Lorrain and White [394] proposed that individuals are role equivalent if they are related to the same individuals.

Definition 9.1.1. *Let $G = (V, E)$ be a graph, and $r\colon V \to W$ a role assignment. Then, r is called* strong structural *if equivalent vertices have the same (out- and in-)neighborhoods, i. e., if for all $u, v \in V$*

$$r(u) = r(v) \Longrightarrow N^+(u) = N^+(v) \text{ and } N^-(u) = N^-(v) \ .$$

Remember Remark 9.0.2: Definitions for role assignments translate to associated partitions and equivalence relations.

Remark 9.1.2. By Definition 9.0.3 it holds for any role assignment r that, if (u, v) is an edge in the graph, then $(r(u), r(v))$ is an edge in the role graph. If r is strong structural, then the converse is also true. This is even an equivalent condition for a role assignment to be strong structural [579]. That is, a role assignment r is strong structural if and only if for all $u, v \in V$, it holds that $(r(u), r(v))$ is an edge in the role graph if and only if (u, v) is an edge in the graph.

We present some examples for strong structural equivalences. The identity mapping $\mathrm{id}\colon V \to V;\ v \mapsto v$ is strong structural for each graph $G = (V, E)$ independent of E. Some slightly less trivial examples are shown in Figure 9.1. For the star, the role assignment that maps the central vertex onto one role and all other vertices onto another, is strong structural. The bipartition of a complete bipartite graph is strong structural. The complete graph without loops has no strong structural role assignment besides id, since the neighborhood of each vertex v is the only one which does not contain v.

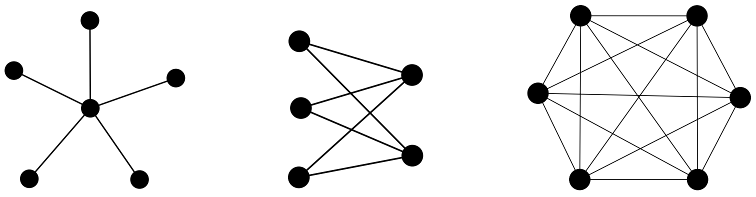
Fig. 9.1. Star (left), complete bipartite graph (middle) and complete graph (right)

We note some elementary properties. A class of strong structurally equivalent vertices is either an independent set (induces a subgraph without edges) for the graph or a clique with all loops. In particular, if two adjacent vertices u, v are strong structurally equivalent, then both (u, v) and (v, u) are edges of the graph, and both u and v have a loop.

The undirected distance of two structurally equivalent (non-isolated) vertices is at most 2. For if u and v are structurally equivalent and u has a neighbor w then w is also a neighbor of v. Thus, structural equivalence can only identify vertices that are near each other.

Although in most irregular graph there won't be any non-trivial structural equivalence, the set of structural equivalences might be huge. For the complete graph with loops, every equivalence is structural. In Section 9.1.2, we investigate a partial order on this set.

Variations of structural equivalence. The requirement that strong structurally equivalent adjacent vertices must have loops has been relaxed by some authors.

Definition 9.1.3 ([191]). *An equivalence $\sim$ on the vertex set of a graph is called* structural *if for all vertices $u \sim v$ the transposition of u and v is an automorphism of the graph.*

White and Reitz [579] gave a slightly different definition, which coincides with Definition 9.1.3 on loopless graphs.

9.1.1 Lattice of Equivalence Relations

The set of equivalence relations on a set V is huge. Here we show that this set naturally admits a partial order, which turns out to be a lattice. (For more on lattice theory, see e. g., [261].) This section is preliminary for Sections 9.1.2 and 9.2.2.

Equivalence relations on a set V are subsets of $V \times V$, thus they can be partially ordered by set-inclusion ($\sim_1 \leq \sim_2$ iff $\sim_1 \subseteq \sim_2$). The equivalence relation $\sim_1$ is then called *finer* than $\sim_2$ and $\sim_2$ is called *coarser* than $\sim_1$. This partial order for equivalences translates to associated partitions and role assignments (see remark 9.0.2).

In partially ordered sets, two elements are not necessarily comparable. In some cases we can at least guarantee the existence of lower and upper bounds.

Definition 9.1.4. *Let X be a set that is partially ordered by $\leq$ and $Y \subseteq X$.*

$y^* \in X$ is called an upper bound *(a* lower bound*) for Y if for all $y \in Y$, $y \leq y^*$ ($y^* \leq y$).*

$y^* \in X$ is called the supremum *(*infimum*) of Y, if it is an upper bound (lower bound) and for each $y' \in X$ that is an upper bound (lower bound) for Y, it follows $y^* \leq y'$ ($y' \leq y^*$). The second condition ensures that suprema and infima (if they exist) are unique.*

The supremum of Y is denoted by $\sup(Y)$ *the infimum by* $\inf(Y)$. *We also write* $\sup(x, y)$ *or* $\inf(x, y)$ *instead of* $\sup(\{x, y\})$ *or* $\inf(\{x, y\})$*, respectively.*

A lattice *is a partially ordered set L, such that for all $a, b \in L$,* $\sup(a, b)$ *and* $\inf(a, b)$ *exist.* $\sup(a, b)$ *is also called the* join *of a and b and denoted by $a \vee b$.* $\inf(a, b)$ *is also called the* meet *of a and b and denoted by $a \wedge b$.*

If $\sim_1$ and $\sim_2$ are two equivalence relations on V, then their intersection (as sets) is the infimum of $\sim_1$ and $\sim_2$. The supremum is slightly more complicated. It must contain all pairs of vertices that are equivalent in either $\sim_1$ or $\sim_2$, but also vertices that are related by a chain of such pairs: The *transitive closure* of a relation $R \subseteq V \times V$ is defined to be the relation $S \subseteq V \times V$, where for all $u, v \in V$

$$\begin{aligned} uSv \Leftrightarrow\ & \exists k \in \mathbb{N},\ \exists w_1, \ldots, w_k \in V \text{ such that} \\ & u = w_1,\ v = w_k, \text{ and } \forall i = 1, \ldots, k-1 \text{ it is } w_i R w_{i+1} \ . \end{aligned}$$

The transitive closure of a symmetric relation is symmetric, the transitive closure of a reflexive relation is reflexive and the transitive closure of any relation is transitive.

It follows that, if $\sim_1$ and $\sim_2$ are two equivalence relations on V, then the transitive closure of their union is the supremum of $\sim_1$ and $\sim_2$.

We summarize this in the following theorem.

Theorem 9.1.5. *The set of equivalence relations is a lattice.*

The interpretation in our context is the following: Given two equivalence relations identifying vertices that play the same role, there exists a uniquely defined smallest equivalence identifying all vertices which play the same role in either one of the two original equivalences. Moreover, there exists a uniquely defined greatest equivalence distinguishing between actors which play a different role in either one of the two original equivalences.

9.1.2 Lattice of Structural Equivalences

It can easily be verified that if $\sim_1$ and $\sim_2$ are two strong structural equivalences for a graph, then so are their intersection and the transitive closure of their union.

Proposition 9.1.6. *The set of strong structural equivalences of a graph is a sublattice of the lattice of all equivalence relations.*

In particular there exist always a maximum structural equivalence (MSE) for a graph.

The property of being strong structural is preserved under refinement:

Proposition 9.1.7. *If $\sim_1 \leq \sim_2$ and $\sim_2$ is a strong structural equivalence, then so is $\sim_1$.*

Although the above proposition is very simple to prove, it is very useful, since it implies that the set of all structural equivalences of a graph is completely described by the MSE. In the next section we present a linear time algorithm for computing the MSE of a graph.

9.1.3 Computation of Structural Equivalences

Computing the maximal strong structural equivalence for a graph $G = (V, E)$ is rather straight-forward. Each vertex $v \in V$ partitions V into 4 classes (some of which may be empty): Vertices which are in $N^+(v)$, in $N^-(v)$, in both, or in none.

The basic idea of the following algorithm 21 is to compute the intersection of all these partitions by looking at each edge at most twice. This algorithm is an adaption of the algorithm of Paige and Tarjan [459, Paragraph 3] (see Section 9.2.3) for the computation of the regular interior, to the much simpler problem of computing the MSE.

The correctness of algorithm 21 follows from the fact that it divides exactly the pairs of vertices with non-identical neighborhoods.

An efficient implementation requires some datastructures, which will be presented in detail since this is a good exercise for understanding the much more complicated algorithm in Section 9.2.3.

Algorithm 21: Computation of the maximal strong structural equivalence (MSE) of a graph

```
Input: a graph $G = (V, E)$
begin
    maintain a partition $\mathcal{P} = \{C_1, \dots, C_k\}$ of $V$, which initially is the complete
    partition $\mathcal{P} = \{V\}$
    // at the end, $\mathcal{P}$ will be the MSE of $G$
    foreach $v \in V$ do
        foreach class $C$ to which a vertex $u \in N^+(v)$ belongs to do
            create a new class $C'$ of $\mathcal{P}$
            move all vertices in $N^+(v) \cap C$ from $C$ to $C'$
            if $C$ has become empty then
                remove $C$ from $\mathcal{P}$
        foreach class $C$ to which a vertex $u \in N^-(v)$ belongs to do
            create a new class $C'$ of $\mathcal{P}$
            move all vertices in $N^-(v) \cap C$ from $C$ to $C'$
            if $C$ has become empty then
                remove $C$ from $\mathcal{P}$
end
```

– A graph $G = (V, E)$ must permit access to the (out-/in-)incidence list of a vertex v in time proportional to the size of this list.
– Scanning all elements of a list must be possible in linear time.
– An edge must permit access to its source and its target in constant time.
– A partition must allow insertion and deletion of classes in constant time.
– A class must allow insertion and deletion of vertices in constant time.
– A vertex must permit access to its class in constant time.

The requirements on partitions and classes are achieved if a partition is represented by a doubly linked list of its classes and a class by a doubly linked list of its vertices.

One refinement step (the outer loop) for a given vertex v is performed as follows.

1. Scan the outgoing edges of v. For each such edge (v, u), determine the class C of u and create an associated block C' if one does not already exist. Move u from C to C'.
2. During the scanning, create a list of those classes C that are split. After the scanning process the list of split classes. For each such class C mark C' as no longer being associated with C and eliminate C if C is now empty.
3. Scan the incoming edges of v and perform the same steps as above.

A loop for a given v runs in time proportional to the degree of v, if v is non-isolated and in constant time else. An overall running time of $\mathcal{O}(|V| + |E|)$ follows, which is also an asymptotic bound for the space requirement.

Conclusion. Structural equivalence is theoretically and computationally very simple. It is much too strict to be applied to irregular networks and only vertices that have distance at most 2, can be identified by a structural equivalence. Nevertheless, structural equivalence is the starting point for many relaxations (see Chapter 10).

9.2 Regular Equivalence

Regular equivalence goes back to the idea of *structural relatedness* of Sailer [501], who proposed that actors play the same role if they are connected to role-equivalent actors – in contrast to structural equivalence, where they have to be connected to identical actors. Regular equivalence has first been defined precisely by White and Reitz in [579]. Borgatti and Everett (e. g., [191]) gave an equivalent definition in terms of colorings (here called role assignments). A coloring is regular if vertices that are colored the same, have the same colors in their neighborhoods. If $r\colon V \to W$ is a role assignment and $U \subseteq V$ then $r(U) := \{r(u)\,;\ u \in U\}$ is called the *role set* of U.

Definition 9.2.1. *A role assignment* $r\colon V \to W$ *is called* regular *if for all* $u, v \in V$

$$r(u) = r(v) \quad \Longrightarrow \quad r(N^+(u)) = r(N^+(v)) \text{ and } r(N^-(u)) = r(N^-(v)) \ .$$

The righthand side equations are equations of sets. There are many more equivalent definitions, (see e. g., [579, 90]).

Regular role assignments are often considered as *the* class of role assignments. The term *regular* is often omitted in literature.

Regular equivalence and bisimulation. Marx and Masuch [408] pointed out the close relationship between regular equivalence, bisimulation, and dynamic logic. A fruitful approach to find good algorithms for regular equivalence is to have a look at the bisimulation literature.

9.2.1 Elementary Properties

In this section we note some properties of regular equivalence relations.

The *identity* mapping $\mathrm{id}\colon V \to V;\ v \mapsto v$ is regular for all graphs. More generally, every structural role assignment is regular.

The next proposition characterizes when the complete partition, which is induced by the constant role assignment $J\colon V \to 1$ is regular. A *sink* is a vertex with zero outdegree, a *source* is one with zero indegree.

Proposition 9.2.2 ([82]). *The complete partition of a graph* $G = (V, E)$ *is regular if and only if* G *contains neither sinks nor sources or* $E = \emptyset$.

Proof. If: If $E = \emptyset$ then the righthand side in definition 9.2.1 is simply $\emptyset = \emptyset$, thus each role assignment is regular. If G has neither sinks nor sources, then, for all $v \in V$, $J(N^+(v)) = J(N^-(v)) = \{1\}$ and the equations in Definition 9.2.1 are satisfied for all $u, v \in V$.

Only if: Suppose $E \neq \emptyset$ and let $v \in V$ be a sink. Since $E \neq \emptyset$ there exists $u \in V$ with non-zero outdegree. But then

$$J(N^+(v)) = \emptyset \neq \{1\} = J(N^+(u)) \ ,$$

but $J(u) = 1 = J(v)$, thus J is not regular. The case of G containing a source is treated analogously. □

The identity and the complete partition are called *trivial* role assignments. The next lemma is formulated in [190] for undirected connected graphs, but it has a generalization to strongly connected (directed) graphs.

Lemma 9.2.3. *Let G be a strongly connected graph. Then in any non-trivial role assignment r of G, neither $\{r(v)\} = r(N^+(v))$ nor $\{r(v)\} = r(N^-(v))$ holds for any vertex v.*

Proof. If for some vertex v it is $\{r(v)\} = r(N^+(v))$, then the same would need to be true for each vertex in $N^+(v)$. Hence each vertex in successive out-neighborhoods would be assigned the same role and since G is strongly connected it follows that $r(V) = \{r(v)\}$ contradicting the fact that the role assignment is non-trivial. The case of $\{r(v)\} = r(N^-(v))$ for some vertex v is handled equally. □

A graph with at least 3 vertices whose only regular role assignments are trivial is called *role primitive.* The existence of directed role primitive graphs is trivial: For every directed path only the identity partition is regular. Directed graphs which have exactly the identity and the complete partition as regular partitions are for example directed cycles of prime length, since every non-trivial regular equivalence induces a non-trivial divisor of the cycle length.

The existence of undirected role primitive graphs is non-trivial.

Theorem 9.2.4 ([190]). *The graph in Figure 9.2 is role primitive.*

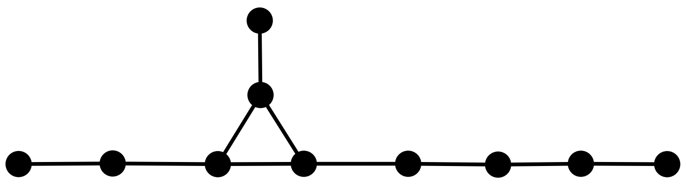

Fig. 9.2. A role-primitive undirected graph

The proof goes by checking that all possible role assignments are either non regular or trivial, where one can make use of the fact that the pending paths of the graph in Figure 9.2 largely diminish the possibilities one has to follow. The proof is omitted here.

A graph in which any role assignment is regular is called *arbitrarily role-assignable.* The next lemma is formulated in [190] for undirected connected graphs.

Lemma 9.2.5. *A strongly connected graph $G = (V, E)$ is arbitrarily role-assignable if and only if it is a complete graph, possibly with some but not necessarily all loops.*

Proof. Let $G = (V, E)$ be a graph satisfying the condition of the lemma and let r be any role assignment. We have to show that for all vertices $u, v \in V$

$$r(u) = r(v) \Longrightarrow r(N^+(u)) = r(N^+(v)) \text{ and } r(N^-(u)) = r(N^-(v)) \ .$$

If $u = v$ this is trivial. Otherwise u and v are connected by a bidirected edge, i.e., the role sets of their in- and out- neighborhoods contain $r(u)$. These role sets also contain all other roles since u and v are connected to all other vertices. So the role sets of the in- and out- neighborhoods of both vertices contain all roles, whence they are equal.

Conversely, let $G = (V, E)$ be a graph with two vertices u and v, such that $u \neq v$ and $(u, v) \notin E$. We assign $V \setminus \{v\}$ one role and v a different one. This is a non-trivial role assignment (note that $n > 2$, since G is connected) with $r(u) = r(N^+(u))$. So by Lemma 9.2.3 this role assignment can't be regular. □

9.2.2 Lattice Structure and Regular Interior

We have seen that the set of regular equivalences of a graph might be huge. In this section we prove that it is a lattice. See the definition of a lattice in Section 9.1.1.

Theorem 9.2.6 ([82]). *The set of all regular equivalences of a graph G forms a lattice, where the supremum is a restriction of the supremum in the lattice of all equivalences.*[1]

Proof. By Lemma 9.2.7, which will be shown after the proof of this theorem, it suffices to show the existence of suprema of arbitrary subsets. The identity partition is the minimal element in the set of regular equivalences, thus it is the supremum for the empty set. Hence we need only to consider the supremum for non-empty collections of regular role assignments. Since the set of all equivalences of a graph is finite, it even suffices to show the existence of the supremum of two regular equivalences.

So let $\sim_1$ and $\sim_2$ be two regular equivalences on G. Define $\equiv$ to be the transitive closure of the union of $\sim_1$ and $\sim_2$.

As mentioned in Section 9.1.1, $\equiv$ is the supremum of $\sim_1$ and $\sim_2$ in the lattice of all equivalences, so it is an equivalence relation and it is a supremum of $\sim_1$

[1] For the infimum see proposition (9.2.9).

and $\sim_2$ with respect to the partial order (which is the same in the lattice of all equivalences and in the lattice of regular equivalences). Therefore it remains to show that $\equiv$ is regular.

For this suppose that $u \equiv v$ and let $x \in N^+(u)$ for $u, v, x \in V$. Since $u \equiv v$ there exists a sequence $u, w_2, \dots, w_{k-1}, v \in V$ where $u \sim_{j_1} w_2$, $j_1 \in \{1, 2\}$. Since $\sim_{j_1}$ is regular and $x \in N^+(u)$, there exists an $x_2 \in V$ such that $x_2 \in N^+(w_2)$ and $x_2 \sim_{j_1} x$. Iterating this will finally produce an x_k such that $x_k \in N^+(v)$ and $x \equiv x_k$, which shows the condition for the out-neighborhood. The case $x \in N^-(u)$ is handled analogously. □

For the proof of Theorem 9.2.6 we need the following lemma (see e. g., [261]).

Lemma 9.2.7. *Let $(X, \leq)$ be a partially ordered set. If $\sup H$ exists for any subset $H \subseteq X$, then $(X, \leq)$ is a lattice.*

Proof. All we have to show is that for $x, y \in X$ there exists $\inf(x, y)$. Let $H := \{z \in X\,;\ z \leq x \text{ and } z \leq y\}$. Then one can easily verify that $\sup H$ is the infimum of $\{x, y\}$. □

Corollary 9.2.8. *If G is a graph then there exists a maximum regular equivalence and there exists a minimum regular equivalence for G.*

Proof. The maximum is simply the supremum over all regular equivalences. Dually, the minimum is the infimum over all regular equivalences. Or easier: The minimum is the identity partition which is always regular and minimal. □

Although the supremum in the lattice of regular equivalences is a restriction of the supremum in the lattice of all equivalences, the infimum is not.

Proposition 9.2.9 ([82]). *The lattice of regular equivalences is not a sublattice of the lattice of all equivalences.*

Proof. We show that the infimum is not a restriction of the infimum in the lattice of all equivalences (which is simply intersection). Consider the graph in Figure 9.3 and the two regular partitions $\mathcal{P}_1 := \{\{A, C, E\}, \{B, D\}\}$ and $\mathcal{P}_2 := \{\{A, C\}, \{B, D, E\}\}$. The intersection of $\mathcal{P}_1$ and $\mathcal{P}_2$ is $\mathcal{P} = \{\{A, C\}, \{B, D\}, \{E\}\}$, which is not regular. □

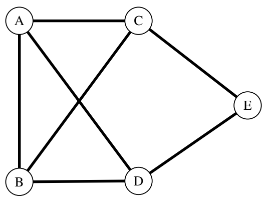

Fig. 9.3. Meet is not intersection

The fact that the supremum in the lattice of regular equivalences is a restriction of the supremum in the lattice of all equivalences implies the existence of a maximum regular equivalence which lies below a given (arbitrary) equivalence.

Definition 9.2.10. *Let G be a graph and $\sim$ an equivalence relation on its vertex set. An equivalence relation $\sim_1$ is called the* regular interior *of $\sim$ if it satisfies the following three conditions.*

1. *$\sim_1$ is regular,*
2. *$\sim_1 \leq \sim$, and*
3. *for all $\sim_2$ satisfying the above two conditions it holds $\sim_2 \leq \sim_1$.*

Corollary 9.2.11. *Let G be a graph and $\sim$ an equivalence relation on its vertex set. Then the regular interior of $\sim$ exists.*

On the other hand there is no minimum regular equivalence above a given equivalence in general (which would have been called a regular closure *or* regular hull*).*

Proof. For the first part, let $G = (V, E)$ be a graph and $\sim$ be an (arbitrary) equivalence on the node set. Then the supremum over the set of all regular equivalence relations that are finer than $\sim$ is the regular interior of $\sim$.

For the second part recall the example in the proof of Prop. 9.2.9 shown in Figure 9.3). It is easy to verify that the regular partitions $\mathcal{P}_1 := \{\{A, C, E\}, \{B, D\}\}$ and $\mathcal{P}_2 := \{\{A, C\}, \{B, D, E\}\}$ are both above the (non-regular) partition $\mathcal{P} := \{\{A, C\}, \{B, D\}, \{E\}\}$ and are both minimal with this property. $\square$

The regular interior is described in more detail in [90]; its computation is treated in Section 9.2.3. The infimum (in the lattice of regular equivalence relations) of two regular equivalence relations $\sim_1$ and $\sim_2$ is given by the regular interior of the intersection of $\sim_1$ and $\sim_2$.

9.2.3 Computation of Regular Interior

The regular interior (see Definition 9.2.10) of an equivalence relation $\sim$ is the coarsest regular refinement of $\sim$. It can be computed, starting with $\sim$, by a number of refinement steps in each of which currently equivalent vertices with non-equivalent neighborhoods are split, until all equivalent vertices have equivalent neighborhoods. For an example of such a computation see Figure 9.4. The running time of this computation depends heavily on how these refinement steps are organized.

In this section we present two algorithms for the computation of the regular interior. CATREGE [83] is the most well-known algorithm in the social network literature. It runs in time $\mathcal{O}(n^3)$. Tarjan and Paige [459] presented a sophisticated algorithm for the *relational coarsest partition problem*, which is essentially equivalent to computing the regular interior. Their algorithm runs in $\mathcal{O}(m \log n)$ time and is well-known in the bisimulation literature. See [408] for the relationship between bisimulation and regular equivalence.

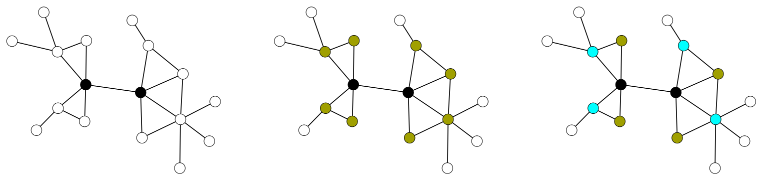

Fig. 9.4. Computation of the regular interior: initial partition (left), first step (middle) second and final step (right)

CATREGE. In [83], Borgatti and Everett proposed CATREGE as an algorithm for computing the maximal regular equivalence of a graph, or more generally for computing the regular interior of an equivalence relation. CATREGE runs in $\mathcal{O}(n^3)$. On a high-level view CATREGE proceeds as follows:

– CATREGE maintains in each refinement step a current partition $\mathcal{P}$, which is initially set to the complete partition (or alternatively to an arbitrary input partition).
– In each refinement step it tests, for each pair of equivalent vertices (w. r. t. $\mathcal{P}$), whether their neighborhoods are equivalent (w. r. t. $\mathcal{P}$). If so, then these vertices remain equivalent, otherwise they will be non-equivalent after this refinement step.
– The algorithm terminates if no changes happen.

The number of refinement steps is bounded by n, since in each refinement step (except the last) the number of equivalence classes grows by at least one. The running time of one refinement step is in $\mathcal{O}(n^2)$.

The Relational Coarsest Partition Problem. This section is taken from [459], although we translate the notation into the context of graphs.

Problem definition. The RELATIONAL COARSEST PARTITION PROBLEM (RCPP) has as input a (directed) graph $G = (V, E)$ and a partition $\mathcal{P}$ of the vertex set V.

For a subset $S \subseteq V$ we write $E(S) := \{v \in V\,;\ \exists u \in S \text{ such that } uEy\}$ and $E^{-1}(S) := \{u \in V\,;\ \exists v \in S \text{ such that } uEy\}$. For two subsets $B \subseteq V$ and $S \subseteq V$, B is called *stable* with respect to S if either $B \subseteq E^{-1}(S)$, or $B \cap E^{-1}(S) = \emptyset$. If $\mathcal{P}$ is a partition of V, $\mathcal{P}$ is called *stable* with respect to S if all of its blocks are stable with respect to S. $\mathcal{P}$ is called *stable* if it is stable with respect to each of its own blocks.

The RCPP is the problem of finding the coarsest stable refinement for the initial partition $\mathcal{P}$.

In the language of role assignments this condition means that for each two roles, say r_1 and r_2, either no vertex, or all vertices assigned r_1 has/have an out-going edge to a vertex assigned r_2. This is the 'out-part' in Definition 9.2.1.

The algorithm of Paige and Tarjan [459] runs in time $\mathcal{O}(m \log n)$ and space $\mathcal{O}(m+n)$. Especially for sparse graphs this is a significant improvement over CATREGE.

Paige and Tarjan already pointed out that it is possible to generalize their algorithm to handle a bounded number of relations. This generalization can be realized in such a way that it yields asymptotically the same running time (see e.g., [207]). Having done this one can apply the algorithm to compute the coarsest stable refinement with respect to E and E^{T} to obtain the regular interior (see Definition 9.2.10).

The SPLIT *function.* The algorithm uses a primitive refinement operation. For each partition $\mathcal{Q}$ of V and subset $S \subseteq V$, let $\textsc{Split}(S, \mathcal{Q})$ be the refinement of $\mathcal{Q}$ obtained by replacing each block B of $\mathcal{Q}$ such that $B \cap E^{-1}(S) \neq \emptyset$ and $B \setminus E^{-1}(S) \neq \emptyset$ by the two blocks $B' := B \cap E^{-1}(S)$ and $B'' := B \setminus E^{-1}(S)$. We call S a *splitter* of $\mathcal{Q}$ if $\textsc{Split}(S, \mathcal{Q}) \neq \mathcal{Q}$. Note that $\mathcal{Q}$ is unstable with respect to S if and only if S is a splitter of $\mathcal{Q}$.

We note the following properties of SPLIT and consequences of stability. Let S and Q be two subsets of V, and let $\mathcal{P}$ and $\mathcal{R}$ be two partitions of V. The following elementary properties are stated without proof.

Property 9.2.12. 1. Stability is *inherited* under refinement; that is, if $\mathcal{R}$ is a refinement of $\mathcal{P}$ and $\mathcal{P}$ is stable with respect to a set S, then so is $\mathcal{R}$.
2. Stability is *inherited* under union; that is, a partition that is stable with respect to two sets is also stable with respect to their union.
3. Function SPLIT is *monotone* in its second argument; that is, if $\mathcal{P}$ is a refinement of $\mathcal{R}$ then $\textsc{Split}(S, \mathcal{P})$ is a refinement of $\textsc{Split}(S, \mathcal{R})$.
4. Function SPLIT is *commutative* in the sense that the coarsest refinement of $\mathcal{P}$ stable with respect to both S and Q is

$$\textsc{Split}(S, \textsc{Split}(Q, \mathcal{P})) = \textsc{Split}(Q, \textsc{Split}(S, \mathcal{P})) \ .$$

Basic algorithm. We begin by describing a naive algorithm for the problem. The algorithm maintains a partition $\mathcal{Q}$ that is initially $\mathcal{P}$ and is refined until it is the coarsest stable refinement. The algorithm consists of repeating the following step until $\mathcal{Q}$ is stable:

REFINE: Find a set S that is a union of some of the blocks of $\mathcal{Q}$ and is a splitter of $\mathcal{Q}$; replace $\mathcal{Q}$ by $\textsc{Split}(S, \mathcal{Q})$.

Some observations. Since stability is inherited under refinement, a given set S can be used as a splitter in the algorithm only once. Since stability is inherited under the union of splitters, after sets are used as splitters their unions cannot be used as splitters. In particular, a stable partition is stable with respect to the union of any subset of its blocks.

Lemma 9.2.13. *The algorithm maintains the invariant that any stable refinement of $\mathcal{P}$ is also a refinement of the current partition $\mathcal{Q}$.*

Proof. By induction on the number of refinement steps. The lemma is true initially by definition. Suppose it is true before a refinement step that refines partition $\mathcal{Q}$ using a splitter S. Let $\mathcal{R}$ be any stable refinement of $\mathcal{P}$. Since S is a union of blocks of $\mathcal{Q}$ and $\mathcal{R}$ is a refinement of $\mathcal{Q}$ by the induction hypothesis, S is a union of blocks of $\mathcal{R}$. Hence $\mathcal{R}$ is stable with respect to S. Since SPLIT is monotone, $\mathcal{R} = \textsc{Split}(S, \mathcal{R})$ is a refinement of $\textsc{Split}(S, \mathcal{Q})$. □

The following theorem gives another proof for the existence of the regular interior (see Corollary 9.2.11).

Theorem 9.2.14. *The refinement algorithm is correct and terminates after at most $n-1$ steps, having computed the unique coarsest stable refinement.*

Proof. The assertion on the number of steps follows from the fact that the number of blocks is between 1 and n. Once no more refinement steps are possible, $\mathcal{Q}$ is stable, and by Lemma 9.2.13 any stable refinement is a refinement of $\mathcal{Q}$. It follows that $\mathcal{Q}$ is the unique coarsest stable refinement. □

The above algorithm is more general than is necessary to solve the problem: There is no need to use unions of blocks as splitters. Restricting splitters to blocks of $\mathcal{Q}$ will also suffice. However, the freedom to split using unions of blocks is one of the crucial ideas needed in developing a fast version of the algorithm.

Preprocessing. In an efficient implementation of the algorithm it it useful to reduce the problem instance to one in which $|E(\{v\})| \geq 1$ for all $v \in V$ (that is only to vertices having out-going edges). To do this we preprocess the partition $\mathcal{P}$ by splitting each block B into $B' := B \cap E^{-1}(V)$ and $B'' := B \setminus E^{-1}(V)$. The blocks B'' will never be split by the refinement algorithm; thus we can run the refinement algorithm on the partition $\mathcal{P}'$ consisting of the set of blocks B'. $\mathcal{P}'$ is a partition of the set $V' := E^{-1}(V)$, of size at most m. The coarsest stable refinement of $\mathcal{P}'$ together with the blocks B'' is the coarsest stable refinement of $\mathcal{P}$. The preprocessing and postprocessing take $\mathcal{O}(m+n)$ time if we have available the preimage set $E^{-1}(v)$ of each element $v \in V$. Henceforth, we shall assume $|E(\{v\})| \geq 1$ for all $v \in V$. This implies $m \geq n$.

Running time of the basic algorithm. We can implement the refinement algorithm to run in time $\mathcal{O}(mn)$ by storing for each element $v \in V$ its preimage set $E^{-1}(v)$. Finding a block of $\mathcal{Q}$ that is a splitter of $\mathcal{Q}$ and performing the appropriate splitting takes $\mathcal{O}(m)$ time. (Obtaining this bound is an easy exercise in list processing.) An $\mathcal{O}(mn)$ time bound for the entire algorithm follows.

Improved algorithm. To obtain a faster version of the algorithm, we need a good way to find splitters. In addition to the current partition $\mathcal{Q}$, we maintain another partition $\mathcal{X}$ such that $\mathcal{Q}$ is a refinement of $\mathcal{X}$ and $\mathcal{Q}$ is stable with respect to every block of $\mathcal{X}$ (in Section 9.3.4, $\mathcal{Q}$ will be called a *relative regular equivalence* w. r. t. $\mathcal{X}$). Initially $\mathcal{Q} = \mathcal{P}$ and $\mathcal{X}$ is the complete partition (containing V as its single block). The improved algorithm consists of repeating the following step until $\mathcal{Q} = \mathcal{X}$:

REFINE: Find a block $S \in \mathcal{X}$ that is not a block of $\mathcal{Q}$. Find a block $B \in \mathcal{Q}$ such that $B \subseteq S$ and $|B| \leq |S|/2$. Replace S within $\mathcal{X}$ by the two sets B and $S \setminus B$; replace $\mathcal{Q}$ by $\text{SPLIT}(S \setminus B, \text{SPLIT}(B, \mathcal{Q}))$.

The correctness of this improved algorithm follows from the correctness of the original algorithm and from the two ways given previously in which a partition can inherit stability with respect to a set.

Special case: If E is a function. Before discussing this algorithm in general, let us consider the special case in which E is a function, i.e., $|E(\{v\})| = 1$ for all $v \in V$. In this case, assume that $\mathcal{Q}$ is a partition stable with respect to a set S that is a union of some of the blocks of $\mathcal{Q}$, and $B \subseteq S$ is a block of $\mathcal{Q}$. Then $\text{SPLIT}(B, \mathcal{Q})$ is stable with respect to $S \setminus B$ as well. This holds, since if B_1 is a block of $\text{SPLIT}(B, \mathcal{Q})$, $B_1 \subseteq E^{-1}(B)$ implies $B_1 \cap E^{-1}(S \setminus B) = \emptyset$, and $B_1 \subseteq E^{-1}(S) \setminus E^{-1}(B)$ implies $B_1 \subseteq E^{-1}(S \setminus B)$. It follows that in each refinement step it suffices to replace $\mathcal{Q}$ by $\text{SPLIT}(B, \mathcal{Q})$, since $\text{SPLIT}(B, \mathcal{Q}) = \text{SPLIT}(S \setminus B, \text{SPLIT}(B, \mathcal{Q}))$. This is the idea underlying Hopcroft's 'process the smaller half' algorithm for the functional coarsest partition problem. The refining set B is at most half the size of the stable set S containing it.

Back to the general case. In the more general relational coarsest partition problem, stability with respect to both S and B does *not* imply stability with respect to $S \setminus B$, and Hopcroft's algorithm cannot be used. This is a serious problem since we cannot afford (in terms of running time) to scan the set $S \setminus B$ in order to perform one refinement step. Nevertheless, we are still able to exploit this idea by refining with respect to both B and $S \setminus B$ using a method that explicitly scans only B.

A preliminary lemma. Consider a general step in the improved refinement algorithm.

Lemma 9.2.15. *Suppose that partition $\mathcal{Q}$ is stable with respect to a set S that is a union of some of the blocks of $\mathcal{Q}$. Suppose also that partition $\mathcal{Q}$ is refined first with respect to a block $B \subseteq S$ and then with respect to $S \setminus B$. Then the following conditions hold:*

1. *Refining $\mathcal{Q}$ with respect to B splits a block $D \in \mathcal{Q}$ into two blocks $D_1 = D \cap E^{-1}(B)$ and $D_2 = D - D_1$ iff $D \cap E^{-1}(B) \neq \emptyset$ and $D \setminus E^{-1}(B) \neq \emptyset$.*
2. *Refining $\text{SPLIT}(B, \mathcal{Q})$ with respect to $S \setminus B$ splits D_1 into two blocks $D_{11} = D_1 \cap E^{-1}(S \setminus B)$ and $D_{12} = D_1 - D_{11}$ iff $D_1 \cap E^{-1}(S \setminus B) \neq \emptyset$ and $D_1 \setminus E^{-1}(S \setminus B) \neq \emptyset$.*
3. *Refining $\text{SPLIT}(B, \mathcal{Q})$ with respect to $S \setminus B$ does not split D_2.*
4. *$D_{12} = D_1 \cap (E^{-1}(B) \setminus E^{-1}(S \setminus B))$.*

Proof. Conditions 1 and 2 follow from the definition of SPLIT.

Condition 3: Form Condition 1 it follows that if D is split, it is $D \cap E^{-1}(B) \neq \emptyset$. Since D is stable with respect to S, and since $B \subseteq S$, then $D_2 \subseteq D \subseteq E^{-1}(S)$. Since by Cond. 1 $D_2 \cap E^{-1}(B) = \emptyset$, it follows that $D_2 \subseteq E^{-1}(S \setminus B)$.

Condition 4: This follows from the fact that $D_1 \subseteq E^{-1}(B)$ and $D_{12} = D_1 \setminus E^{-1}(S \setminus B)$. $\square$

Performing the three-way splitting of a block D into D_{11}, D_{12}, and D_2 as described in Lemma 9.2.15 is the hard part of the algorithm. Identity 4 of Lemma 9.2.15 is the crucial observation that we shall use in our implementation. Remember that scanning the set $S \setminus B$ takes (possibly) too long to obtain the claimed running time. We shall need an additional datastructure to determine $D_1 \setminus E^{-1}(S \setminus B) = (D \cap E^{-1}(B)) \setminus E^{-1}(S \setminus B)$ by scanning only B.

Running time of the improved algorithm. A given element of V is in at most $\log_2 n + 1$ different blocks B used as refining sets, since each successive such set is at most half the size of the previous one. We shall describe an implementation of the algorithm in which a refinement step with respect to block B takes $\mathcal{O}(|B| + \sum_{u \in B} |E^{-1}(\{u\})|)$ time. From this an $\mathcal{O}(m \log n)$ overall time bound for the algorithm follows by summing over all blocks B used for refinement and over all elements in such blocks.

Datastructures. (See Section 9.1.3 for an example of a much simpler algorithm which already uses some of the ideas of this algorithm.)

Graph $G = (V, E)$ is represented by the sets V and E. Partitions $\mathcal{Q}$ and $\mathcal{X}$ are represented by doubly linked lists of their blocks.

A block S of $\mathcal{X}$ is called *simple* if it contains only a single block of $\mathcal{Q}$ (equal to S but indicated by its own record) and *compound* if it contains two or more blocks of $\mathcal{Q}$.

The various records are linked together in the following ways. Each edge uEv points its source u. Each vertex v points to a list of incoming edges uEv. This allows scanning the set $E^{-1}(\{v\})$ in time proportional to its size. Each block of $\mathcal{Q}$ has an associated integer giving its size and points to a doubly linked list of the vertices in it (allowing deletion in $\mathcal{O}(1)$ time). Each vertex points to the block of $\mathcal{Q}$ containing it. Each block of $\mathcal{X}$ points to a doubly linked list of the blocks of $\mathcal{Q}$ contained in it. Each block of $\mathcal{Q}$ points to the block of $\mathcal{X}$ containing it. We also maintain a set C of compound blocks of $\mathcal{X}$. Initially C contains the single block V, which is the union of the blocks of $\mathcal{P}$. If $\mathcal{P}$ contains only one block (after the preprocessing), $\mathcal{P}$ itself is the coarsest stable refinement and we terminate the algorithm here.

To make three-way splitting (see Lemma 9.2.15) fast we need one more collection of records. For each block S of $\mathcal{X}$ and each element $v \in E^{-1}(S)$ we maintain an integer $\textsc{Count}(v, S) := |S \cap E(\{v\})|$. Each edge uEv with $v \in S$ contains a pointer to $\textsc{Count}(u, S)$. Initially there is one count per vertex (i. e., $\textsc{Count}(v, V) = |E(\{v\})|$) and each edge uEv points to $\textsc{Count}(u, V)$.

This $\textsc{Count}$ function will help to determine the set $E^{-1}(B) \setminus E^{-1}(S \setminus B)$ in time proportional to $|\{uEv\,;\ v \in B\}|$ (see step 5 below).

Both the space needed for all the data structures and the initialization time is $\mathcal{O}(m)$.

The refinement algorithm consists of repeating refinement steps until C is empty.

Performing one refinement step. For clarity we divide one refinement step into 7 substeps.
1. (select a refining block). Remove some block S from C. (Block S is a compound block of $\mathcal{X}$.) Examine the first two blocks in the list of blocks of $\mathcal{Q}$ contained in S. Let B be the smaller one. (Break a tie arbitrarily.)
2. (update $\mathcal{X}$). Remove B from S and create a new (simple) block S' of $\mathcal{X}$ containing B as its only block of $\mathcal{Q}$. If S is still compound, put S back into C.
3. (compute $E^{-1}(B)$). Copy the vertices of B into a temporary set B'. (This facilitates splitting B with respect to itself during the refinement.) Compute $E^{-1}(B)$ by scanning the edges uEv such that $v \in B$ and adding each vertex u in such an edge to $E^{-1}(B)$ if it has not already been added. Duplicates are suppressed by marking vertices as they are encountered and linking them together for later unmarking. During the same scan compute $\textsc{Count}(u, B) = |\{v \in B\, ;\; uEv\}|$, store this count in a new integer and make u point to it. These counts will be used in step 5.
4. (refine $\mathcal{Q}$ with respect to B). For each block D of $\mathcal{Q}$ containing some element (vertex) of $E^{-1}(B)$, split D into $D_1 = D \cap E^{-1}(B)$ and $D_2 = D \setminus D_1$. Do this by scanning the elements of $E^{-1}(B)$. To process an element $u \in E^{-1}(B)$, determine the block D of $\mathcal{Q}$ containing it and create an associated block D' if one does not already exist. Move u from D to D'.

During the scanning, construct a list of those blocks D that are split. After the scanning, process the list of split blocks. For each such block D with associated block D', mark D' as no longer being associated with D (so that it will be correctly processed in subsequent iterations of Step 4). Eliminate the record for D if D is now empty and, if D is nonempty and the block of $\mathcal{X}$ containing D and D' has been made compound by the split, add this block to C.
5. (compute $E^{-1}(B)\setminus E^{-1}(S\setminus B)$). Scan the edges uEv with $v \in B'$. To process an edge uEv, determine $\textsc{Count}(u, B)$ (to which u points) and $\textsc{Count}(u, S)$ (to which uEv points). If $\textsc{Count}(u, B) = \textsc{Count}(u, S)$, add u to $E^{-1}(B) \setminus E^{-1}(S \setminus B)$ if it has not been added already.
6. (refine $\mathcal{Q}$ with respect to $S \setminus B$). Proceed exactly as in Step 4 but scan $E^{-1}(B) \setminus E^{-1}(S \setminus B)$ (computed in Step 5) instead of $E^{-1}(B)$.
7. (update counts). Scan the edges uEv such that $v \in B'$. To process and edge uEv, decrement $\textsc{Count}(u, S)$ (to which uEv points). If this count becomes zero, delete the $\textsc{Count}$ record, and make uEv point to $\textsc{Count}(u, B)$ (to which u points). After scanning all the appropriate edges, discard B'.

Note that in step 5 only edges terminating in B' are scanned. Step 5 is correct (computes $E^{-1}(B) \setminus E^{-1}(S \setminus B)$) since for each vertex u in $E^{-1}(B)$, it holds that u is in $E^{-1}(B) \setminus E^{-1}(S \setminus B)$ iff u is not in $E^{-1}(S \setminus B)$ iff all edges starting at u and terminating in S terminate in B iff $\textsc{Count}(u, B) = \textsc{Count}(u, S)$.

The correctness of this implementation follows in a straightforward way from our discussion above of three-way splitting. The time spent in a refinement step is $\mathcal{O}(1)$ per edge terminating in B plus $\mathcal{O}(1)$ per vertex of B, for a total of $\mathcal{O}(|B| + \sum_{v \in B} |E^{-1}(\{v\})|)$ time. An $\mathcal{O}(m \log n)$ time bound for the entire algorithm follows as discussed above. It is possible to improve the efficiency of the algorithm by a constant factor by combining various steps, which have been kept separate for clarity.

Adaptation to Related Problems. The above algorithm turns out to be the key to efficiently solve several partition refinement problems that arise in this chapter. We will briefly sketch this generality.

Computing the maximal strong structural equivalence (as described in Section 9.1.3) or the relative regular equivalence (see Section 9.3.4) is much simpler than computing the regular interior. Nevertheless we can use the idea of iteratively splitting blocks according to intersection with certain neighborhoods. (See algorithm 21 and the comments in Section 9.3.4.) These problems can be solved by algorithms that run in $\mathcal{O}(m + n)$.

Computing the coarsest *equitable* (see Section 9.3.1) has been solved earlier than the problem of computing the regular interior (see [110] for an $\mathcal{O}(m \log n)$ algorithm and the comments in [459]).

Refining a partition w. r. t. *multiple relations* (see Definition 9.4.1) is also possible in $\mathcal{O}(m \log n)$ (if the number of relations is bounded by a constant). This extension of the algorithm can be used to compute the regular interior w. r. t. incoming and out-going edges. Shortly, a partition can be refined w. r. t. multiple relations by performing steps 3–7 (see above) for fixed B and S successively for all relations, one at a time. (See e. g., [207].)

9.2.4 The Role Assignment Problem

In this section we investigate the computational complexity of the decision problem whether a given graph admits a regular role assignment with prespecified role graph, or with prespecified number of equivalence classes. In this section we consider only undirected graphs.

The most complete characterization is from Fiala and Paulusma [209]. Let $k \in \mathbb{N}$ and R be an undirected graph, possibly with loops.

Problem 9.2.16 (k-Role Assignment (k-RA)). Given a graph G.
Question: Is there a regular equivalence for G with exactly k equivalence classes?

Problem 9.2.17 (R-Role Assignment (R-RA)). Given a graph G.
Question: Is there a regular role assignment $r : V(G) \to V(R)$ with role graph R?

Note that we require role assignments to be surjective mappings.

Theorem 9.2.18 ([209]). *k-RA is polynomially solvable for $k = 1$ and it is $\mathcal{NP}$-complete for all $k \geq 2$.*

Theorem 9.2.19 ([209]). *R-RA is polynomially solvable if each component of R consists of a single vertex (with or without a loop), or consists of two vertices without loops and it is $\mathcal{NP}$-complete otherwise.*

We give the proof of one special case of the R-Role Assignment Problem.

Theorem 9.2.20 ([493]). *Let R_0 be the graph in Figure 9.5. Then R_0-RA is $\mathcal{NP}$-complete.*

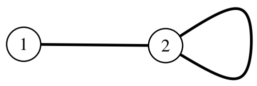

Fig. 9.5. Role graph R_0

Proof. It is easy to see that R-RA is in $\mathcal{NP}$ since one can easily check in polynomial time whether a given function $r : V \to \{1, 2\}$ is a 2-role assignment with role graph R_5.

We will show that the 3-satisfiability problem (3SAT) is polynomially transformable to R_0-RA. So let $U = \{u_1, \ldots, u_n\}$ be a set of variables and $C = \{c_1, \ldots, c_m\}$ be a set of clauses (each consisting of exactly three literals). We will construct a graph $G = (V, E)$ such that G is 2-role assignable with role graph R_0 if and only if C is satisfiable.

The construction will be made up of two components, truth-setting components and satisfaction testing components (see Figure 9.6).

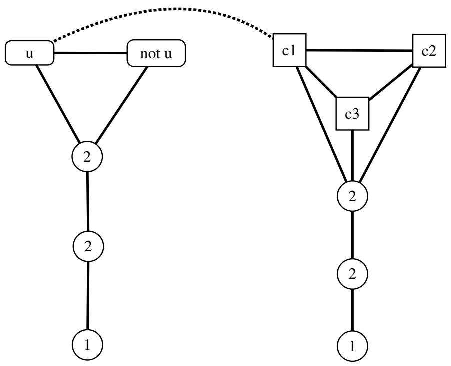

Fig. 9.6. Truth-setting component for variable u (left); satisfaction testing component for clause $\{c_1, c_2, c_3\}$ (right) and communication edge if literal c_1 equals u (dashed). The roles of the vertices in the pending paths are uniquely determined (as indicated by the labels 1 resp. 2) if the role assignment should be regular with role graph R_0

For each variable $u_i \in U$, there is a truth-setting component $T_i = (V_i, E_i)$ with

$$\begin{aligned} V_i &:= \{u_i, \overline{u_i}, a_{i1}, a_{i2}, a_{i3}\} \ , \\ E_i &:= \{u_i\overline{u_i}, u_i a_{i3}, \overline{u_i} a_{i3}, a_{i1}a_{i2}, a_{i2}a_{i3}\} \ . \end{aligned}$$

Note that, although we write $u_i\overline{u_i}$ for the edge $\{u_i, \overline{u_i}\}$, the graph is undirected.

The intuition behind the construction of T_i is the following: If a graph containing T_i as a subgraph (such that the a_{ij} are adjacent only to the vertices in V_i as specified above) admits a regular role assignment r with role graph R_0, then necessarily $r(a_{i1}) = 1$, since a_{i1} has degree one and a vertex which is assigned 2 must have degree ≥ 2. Then $r(a_{i2}) = 2$, since a 1-vertex is adjacent to a 2-vertex and $r(a_{i2}) = 2$, since a 2-vertex is adjacent to a 2-vertex. Finally exactly one of u_i or $\overline{u_i}$ is assigned 2, meaning that variable u_i is set to *true* or *false*, respectively. Thus component T_i ensures that a variable gets either *true* or *false*.

For each clause $c_j \in C$, let vertices c_{j1}, c_{j2}, and c_{j3} be three vertices corresponding to the three literals in the clause c_j. Then there is a satisfaction testing component $S_j = (V_j', E_j')$ with

$$\begin{aligned} V_j' &:= \{c_{j1}, c_{j2}, c_{j3}, b_{j1}, b_{j2}, b_{j3}\} \ , \\ E_j' &:= \{c_{j1}c_{j2}, c_{j1}c_{j3}, c_{j2}c_{j3}, c_{j1}b_{j3}, c_{j2}b_{j3}, c_{j3}b_{j3}, b_{j1}b_{j2}, b_{j2}b_{j3}\} \ . \end{aligned}$$

The intuition behind the construction of S_j is the following: If a graph containing S_j as a subgraph (such that the b_{jl} are adjacent only to the vertices in V_j as specified above) admits a regular role assignment r with role graph R_0, then necessarily $r(b_{j1}) = 1$, $r(b_{j2}) = r(b_{j3}) = 2$, which ensures that one of the vertices c_{j1}, c_{j2}, c_{j3} is assigned 1, thus ensuring that every adjacent vertex of this 1-vertex must be assigned 2. This will be crucial later.

The construction so far is only dependent on the number of variables and clauses. The only part of the construction that depends on which literals occur in which clauses is the collection of communication edges. For each clause $c_j = \{x_{j1}, x_{j2}, x_{j3}\} \in C$ the communication edges emanating from S_j are given by

$$E_j'' := \{c_{j1}x_{j1}, c_{j2}x_{j2}, c_{j3}x_{j3}\} \ .$$

(The x_{jl} are either variables in U or their negations.) Notice that for each c_{jk}, there is exactly one vertex that is adjacent to c_{jk} in E_j'', which is the corresponding literal vertex for c_{jk} in the clause c_j.

To complete the construction of our instance of R_0-RA, let $G = (V, E)$ with V being the union of all V_is and all V_j's and E the union of all E_is, all E_j's and all E_j''s.

As mentioned above, given a regular role assignment for G with role graph R_0, for each $j = 1, \ldots, m$ there is a vertex c_{jk} such that $r(c_{jk}) = 1$ implying that the corresponding adjacent literal is assigned 2. Setting this literal to *true* will satisfy clause c_j.

Thus we have shown that the formula is satisfiable if G is regularly R_0 assignable.

Conversely, suppose that C has a satisfying truth assignment. We obtain an assignment $r \colon V \to \{1, 2\}$ as follows. For each $i = 1, \ldots, n$ set $r(u_i)$ to 2 (and

$r(\overline{u_i})$ to 1) if and only if variable u_i is *true* and set the role of the vertices a_{ik} and b_{jk} as implied by the fact that r should be regular (see above). Moreover, for each $j = 1, \ldots, m$ let c_{jk}, $k \in \{1,2,3\}$, be some vertex whose corresponding literal in the clause c_j is *true* – such a k exists since the truth assignment is satisfying for C. Set $r(c_{jk}) := 1$ and $r(c_{jl}) := 2$ for $l \in \{1,2,3\}$, $l \neq k$.

The proof is complicated a bit by the fact that more than one literal in a clause might be true, but setting $r(c_{jk}) = 1$ is allowed for only one $k \in \{1,2,3\}$. Since a 2-vertex may be adjacent to another 2-vertex, this does not destroy the regularity of r. □

9.2.5 Existence of k-Role Assignments

We have seen in the previous section that the decision whether a graph admits a regular equivalence with exactly k equivalence classes is $\mathcal{NP}$-complete for general graphs. Nevertheless, there are easy-to-verify sufficient, if not necessary, conditions that guarantee the existence of regular k-role assignments. Briefly, the condition is that the graph differs not too much from a regular graph.

Theorem 9.2.21 ([474]). *For all $k \in \mathbb{N}$ there is a constant $c_k \in \mathbb{R}$ such that for all graphs G with minimal degree $\delta = \delta(G)$ and maximal degree $\Delta = \Delta(G)$ satisfying*

$$\delta \geq c_k \log(\Delta) \ ,$$

there is a regular equivalence for G with exactly k equivalence classes.

To exclude trivial counterexamples we assume in the following that all graphs in question have at least k vertices.

For the proof we need a uniform version of the Lovasz Local Lemma.

Theorem 9.2.22 ([25, Chapter 5 Corollary1.2]). *Let $A_i, i \in I$, be events in a discrete probability space. If there exists M such that for every $i \in I$*

$$|\{A_j \,;\, A_j \text{ is not independent of } A_i\}| \leq M \ ,$$

and if there exists $p > 0$ such that $\Pr(A_i) \leq p$ for every $i \in I$, then

$$ep(M+1) \leq 1 \Longrightarrow \Pr\left(\bigcap_{i \in I} \overline{A_i}\right) > 0 \ ,$$

where e is the Euler *number $e = \sum_{i=0}^{\infty} 1/i!$.* □

Proof (of Theorem 9.2.21). Define $r : V \to \{1, \ldots, k\}$ as follows: For every $v \in V$ choose $r(v)$ uniformly at random from $\{1, \ldots, k\}$.

For $v \in V$, let A_v be the event that $r(N(v)) \neq \{1, \ldots, k\}$. It is

$$\Pr(A_v) \leq k \left(\frac{k-1}{k}\right)^{d(v)} \leq k \left(\frac{k-1}{k}\right)^{\delta(G)} \ .$$

Because all $r(w)$ are chosen independently and for a fixed value i, the probability that i is not used for any of the vertices adjacent to v is $\left(\frac{k-1}{k}\right)^{d(v)}$, and there are k choices for i.

Also note that A_v and A_w are not independent if and only if $N(v) \cap N(w) \neq \emptyset$. Hence, A_v with $M := \Delta(G)^2$ and $p := k\left(\frac{k-1}{k}\right)^{\delta(G)}$ satisfies the conditions of the Lovasz Local Lemma. Therefore,

$$ek\left(\frac{k-1}{k}\right)^{\delta(G)}(\Delta(G)^2+1) \leq 1 \Rightarrow \Pr\left(\bigcap_{v\in V}\overline{A_v}\right) > 0 \ . \tag{9.1}$$

If the righthand side of (9.1) holds, there exists at least one r such that $r(N(v)) = \{1,\ldots,k\}$ for every $v \in V$, that is, there exists at least one regular k-role assignment. In order to finish the proof we note that the lefthand side of (9.1) is equivalent to

$$\delta(G) \geq \frac{\log(ek(\Delta(G)^2+1))}{\log\left(\frac{k}{k-1}\right)} \ .$$

Clearly, there exists a constant c_k such that $c_k \log(\Delta(G))$ is greater than the righthand side of the above inequality. □

Conclusion. Regular equivalences are well investigated in computer science. Results indicate that many regular equivalences exist even in irregular graphs, but it is unclear how to define and/or compute the best, or at least a good one. Fast algorithms exist for the computation of the maximal regular equivalence or for the regular interior of an a priori partition. The maximal regular equivalence could be meaningful for directed graphs (for undirected it is simply the division into isolates and non-isolates). Also, the regular interior could be a good role assignment if one has an idea for the partition to be refined. Specifying the number of equivalence classes or the role graph yields $\mathcal{NP}$-hard problems, in the general case. Optimization approaches for these problems are presented in Section 10.1.7 in the next chapter.

9.3 Other Equivalences

In this section we briefly mention other (than structural or regular) types of role equivalences.

9.3.1 Exact Role Assignments

In this section we define a class of equivalence relations that is a subset of regular equivalences. These equivalences will be called *exact*. The associated partitions are also known as *equitable partitions* in graph theory, they have first been defined as *divisors* of graphs.

While for regular equivalences only the occurrence or non-occurrence of a role in the neighborhood of a vertex matters, for exact equivalences, the number of occurrence matters.

The graph model of this section are undirected multigraphs.

Definition 9.3.1. *A role assignment r is called* exact *if for all $u, v \in V$*

$$r(u) = r(v) \quad \Longrightarrow \quad r(N(u)) = r(N(v)) \ ,$$

where the last equation is an equation of multi-sets*, i. e., vertices, that have the same role, must have the same number of each of the other roles in their neighborhoods.*

The coloring in Figure 9.7 defines an exact role assignment for the shown graph.

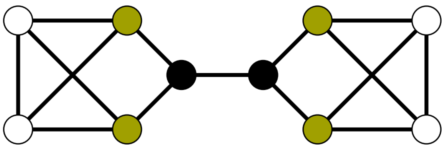

Fig. 9.7. An exact role assignment

While an equivalence is regular for a multigraph if and only if it is regular for the induced simple graph (each edge at most once), for exact equivalences the multiplicity of an edge matters.

It is straightforward to see that exact role assignments are regular, the converse is not true.

An equivalent definition is the following.

Definition 9.3.2 ([247]). *A partition $\mathcal{P} = \{C_1, \dots, C_k\}$ of the vertex set V of an undirected (multi-)graph $G = (V, E)$ is called* equitable *if there are integers b_{ij}, $i, j = 1, \dots, k$, such that each vertex in class C_i has exactly b_{ij} neighbors in class C_j. The matrix $B = (b_{ij})_{i,j=1,\dots,k}$ defines a (directed) multi-graph, which is called the* quotient *of G modulo $\mathcal{P}$, denoted by $G/\mathcal{P}$.*

A partition is equitable if and only if the associated role assignment is exact. The above definition also extends the definition of the quotient or role graph (see Section 9.0.2) to multigraphs. Note that this is possible only for exact role assignments.

Note that even if the graph is undirected the quotient is possibly directed, meaning that the multiplicity of an edge may differ from the multiplicity of the reversed edge. This happens always if two 'adjacent' equivalence classes are of different size.

Exact role assignments are compatible with algebraic properties of a graph.

Theorem 9.3.3 ([247]). *Let G be a graph, $\mathcal{P}$ an equitable partition. Then, the characteristic polynomial of the quotient $G/\mathcal{P}$ divides the characteristic polynomial of G.* □

This theorem implies that the spectrum of the quotient $G/\mathcal{P}$ is a subset of the spectrum of G.

The set of all exact role assignments of a graph forms a lattice [191]. The maximal exact role assignment of a graph can be computed by an adaption of the algorithm in Section 9.2.3. (See [110] and the comments in [459].)

Many problems around exact role assignments are $\mathcal{NP}$-complete as well. For example the problem of deciding if a graph G admits an exact role assignment with quotient R is $\mathcal{NP}$-complete if both G and R are part of the input, or for some fixed R. This holds, since the $\mathcal{NP}$-complete problem of deciding whether a 3-regular graph has a perfect code [370], can be formulated as the problem of deciding whether G has an exact role assignment with quotient

$$R = \begin{bmatrix} 0 & 3 \\ 1 & 2 \end{bmatrix} .$$

The quotient over an equitable partition has much more in common with the original graph than, e. g., the role graph over a regular equivalence. Exact role assignments also ensure that equivalent vertices have the same degree, which is not true for regular role assignments.

Conclusion. Exact role assignments, also called equitable partitions are well investigated in algebraic graph theory. While some problems around equitable partitions are $\mathcal{NP}$-complete, there are efficient algorithms to compute the maximal equitable partition of a graph, or to compute the coarsest equitable refinement of an a priori partition. These algorithms could be used to compute role assignments, but, due to irregularities, the results contain in most cases too many classes and miss the underlying (possibly perturbed) structure. Brandes and Lerner [97] introduced a relaxation of equitable partitions that is tolerant against irregularities.

9.3.2 Automorphic and Orbit Equivalence

Automorphic equivalence expresses interchangeability of vertices.

Definition 9.3.4 ([191]). *Let $G = (V, E)$ be a graph, $u, v \in V$. Then u and v are said to be* automorphically equivalent *if there is an automorphism φ of G with $\varphi(u) = v$.*

Automorphically equivalent vertices cannot be distinguished only in terms of the graph structure. Therefore it could be argued that at least automorphically equivalent vertices should be considered to play the same role.

It is easy to see that structurally equivalent vertices are automorphically equivalent.

A partition of the vertex set which has the property that each pair of equivalent vertices is automorphically equivalent is not necessarily a regular equivalence. However we have the following result.

Proposition 9.3.5 ([190]). *Let $G = (V, E)$ be a graph with automorphism group $A(G)$, and $H < A(G)$ be a subgroup of $A(G)$. Then assigning roles according to the orbits of H defines an exact role assignment for G. Such a partition is called an* orbit partition.

Proof. Let r be a role assignment as in the formulation of the proposition. If $r(u) = r(v)$ then there exists $\varphi \in H$ such that $\varphi(u) = v$. If $x \in N^+(u)$, then $\varphi(x) \in N^+(\varphi(u)) = N^+(v)$. Furthermore $r(x) = r(\varphi(x))$ by definition. It follows that $r(N^+(u)) \subseteq r(N^+(v))$ (as multisets). The other inclusion and the corresponding assertion for the in-neighborhoods is shown similar. □

In particular, orbit equivalences are regular.

For example, the coloring in Figure 9.7 defines the orbit partition of the automorphism group of the shown graph.

The set of orbit equivalences forms a proper subset of the set of all exact equivalences, which can be proved by any regular graph which is not vertex-transitive. For example, the complete partition for the graph in Figure 9.7 is exact but not an orbit partition.

The above proposition can also be used to prove that every undirected role primitive graph (see Section9.2.1) is a graph with trivial automorphism group [190]. This is not true for directed graphs as can be seen by directed cycles of prime length.

Orbit equivalence has the nice feature that its condition is invariant w. r. t. a shift to the complement graph. This does not hold neither for regular nor for exact equivalence.

The computation of orbit equivalences is related to the problem of computing the automorphism group which has open complexity status.

Conclusion. Automorphically equivalent vertices cannot be distinguished in terms of graph structure, but only by additional labels or attributes. It could therefore be argued that at least automorphically equivalent vertices play the same role. Computation of automorphic equivalence seems to be hard, but, in irregular networks, there won't be any significant automorphisms anyway.

9.3.3 Perfect Equivalence

Perfect equivalence is a restriction of regular equivalence. It expresses the idea that there must be a reason for two vertices for being *not* equivalent.

Definition 9.3.6 ([191]). *A role assignment r defines a* perfect equivalence *if for all $u, v \in V$*

$$r(u) = r(v) \quad \Longleftrightarrow \quad r(N^+(u)) = r(N^+(v)) \text{ and } r(N^-(u)) = r(N^-(v)).$$

A regular equivalence is perfect if and only if the induced role graph has no strong structural equivalent vertices (see Section 9.1).

The set of perfect equivalence relations of a graph is a lattice [191], which is neither a sublattice of all equivalence relations (Section 9.1.1) nor of the lattice of regular equivalence relations (Section 9.2.2). A *perfect interior* of an equivalence relation $\sim$ would be a coarsest perfect refinement of $\sim$ (compare Definition 9.2.10). In contrast to the regular interior, the perfect interior does not exist in general.

Theorem 9.3.7. *In general, the transitive closure (see Section 9.1.1) of the union of two perfect equivalence relations is not perfect. In particular, for some equivalences there is no perfect interior.*

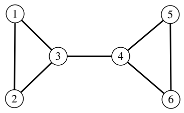

Fig. 9.8. Graph for the proof of Theorem 9.3.7. Supremum of two perfect equivalences is not perfect

Proof. Consider the graph in Figure 9.8 and the two perfect partitions $\mathcal{P}_1 = \{\{1,5\},\{2,6\}\{3,4\}\}$ and $\mathcal{P}_2 = \{\{1,2\},\{5,6\}\{3\},\{4\}\}$. The transitive closure of $\mathcal{P}_1$ and $\mathcal{P}_2$ is $\mathcal{P} = \{\{1,2,5,6\},\{3,4\}\}$, which is not perfect.

For the second statement, note that $\mathcal{P}_1$ and $\mathcal{P}_2$ are both perfect refinements of $\mathcal{P}$ and are both maximal w.r.t. this property. □

The second statement has a more trivial proof: For a graph with two strong structurally equivalent vertices, the identity partition has no perfect refinement.

Some decision problems concerning perfect equivalence are $\mathcal{NP}$-complete as well. This can be seen by Theorems 9.2.18 and 9.2.19, restricted to role graphs without strong structurally equivalent vertices.

Although perfect equivalences rule out some trivial regular equivalences, there is no evidence why roles shouldn't be strong structurally equivalent.

Conclusion. Perfect equivalence is a restriction of regular equivalence, but it doesn't seem to yield better role assignments. Some mathematical properties of regular equivalences get lost and there are examples where the condition on perfect equivalence rules out good regular role assignments.

9.3.4 Relative Regular Equivalence

Relative regular equivalence expresses the idea that equivalent vertices have equivalent neighborhoods in a coarser, predefined measure.

Definition 9.3.8 ([90]). *Let $G = (V, E)$ be a graph and $r\colon V \to W$ and $r_0\colon V \to W_0$ be two role assignments. Then, r is called* regular relative *to r_0 if $r \le r_0$ (see Section 9.1.1 for the partial order on the set of role assignments) and for all $u, v \in V$*

$$r(u) = r(v) \Rightarrow r_0(N^+(u)) = r_0(N^+(v)) \text{ and } r_0(N^-(u)) = r_0(N^-(v)) \ .$$

A typical application [90] of relative regular equivalence is given by a network of symmetric friendship ties which a priori is divided into two disjoint friendship cliques A and B. Assume that within each clique every member has at least one tie to some other member of the same clique. The partition into these two cliques would be regular if either there is no tie between the two cliques or each actor would have, in addition to the intra-group ties, at least one tie to a member of the other group. But lets assume that some, but not all, actors have friendship ties to members of the other group. The partition into A and B is no longer regular. Now we can split each group into those actors having ties to some member of the other group and those who don't. Say we obtain the partition into A_1, A_2, B_1, and B_2. Neither is this partition (in general) regular: There might be some actors in, say, A_1 having intra-group ties only with members of A_1, some only with members of A_2, some with both; they don't have equivalent neighborhoods. But they have equivalent neighborhoods with respect to the coarse partition into A and B. Thus, the partition into A_1, A_2, B_1 and B_2 is regular relative to the partition into A and B.

Relative regularity below a fixed equivalence is preserved under refinement. (Compare Prop. 9.1.7 for a similar proposition for structural equivalence.)

Proposition 9.3.9. *Let $\sim$, $\sim_1$, and $\sim_2$ be equivalence relations on V such that $\sim_1 \le \sim_2$ and $\sim_2$ is regular relative to $\sim$. Then so is $\sim_1$.*

Similar to Prop. 9.1.7, this proposition implies that the set of equivalences that are regular relative to a fixed equivalence $\sim$ is a sublattice of all equivalences and is completely described by the maximum of this set, denoted here by MRRE($\sim$).

Computing the MRRE($\sim$) is possible in linear time by an adaptation of the algorithm 21 for computing the maximal structural equivalence: Instead of splitting equivalence classes from the point of view of single vertices, classes are split from the point of view of the classes of $\sim$ (compare the algorithm in Section 9.2.3). Note that the classes of $\sim$ are fixed and the MRRE($\sim$) has been found after all classes of $\sim$ have been processed once.

Each refinement step in the CATREGE algorithm (see Section 9.2.3) computes an equivalence that is regular relative to the previous one, but the running time of one step is in $\mathcal{O}(n^2)$, which is worse than the above described algorithm on sparse graphs.

Conclusion. Relative regular equivalence is computationally simple but it needs an a priori partition of the vertices and, since its compatibility requirement is only local, is not expected to represent global network structure. It has most been applied in connection with multiple and composite relations (see, e. g., Winship-Pattison Role Equivalence in Section 9.5.1).

9.4 Graphs with Multiple Relations

Actors in a social network are often connected by more than one relation. For example, on the set of employees of a company there might be two relations GivesOrdersTo and IsFriendOf. It is often insufficient to treat these relations separately one at a time since their interdependence matters.

In this section we generalize the graph model to graphs with multiple relations, that is, collections of graphs with common vertex set.

Definition 9.4.1. *A* graph with multiple relations $\mathcal{G} = (V, \mathcal{E})$ *consists of a finite vertex set* V*, and a finite set of relations (finite set of edge sets)* $\mathcal{E} = \{E_i\}_{i=1,\ldots,p}$*, where* $p \in \mathbb{N}$ *and* $E_i \subseteq V \times V$*.*

For the remainder of this section we often write 'graph' meaning 'graph with multiple relations'. A graph is identified with the one resulting from deleting duplicate relations, where we say that two relations are equal if they consist of the same pairs of vertices. That is relations don't have 'labels' but are distinguished by the pairs of vertices they contain.

The role graph of a graph with multiple relations is again a graph with (possibly) multiple relations. (Compare Definition 9.0.3 of the role graph of a graph with one relation.)

Definition 9.4.2. *Let* $\mathcal{G} = (V, \mathcal{E})$ *be a graph with multiple relations, and* $r \colon V \to W$ *be a role assignment. The* role graph *of* $\mathcal{G}$ *over* r *is the graph* $\mathcal{R} = (W, \mathcal{F})$*, where* $\mathcal{F} = \{F_i\,;\ i = 1, \ldots, p\}$*, where* $F_i = \{(r(u), r(v))\,;\ (u, v) \in E_i\}$*.*

Note that F_i may be equal to F_j even if $E_i \neq E_j$ and that duplicate edge relations are eliminated ($\mathcal{F}$ is a set).

From the above definition we can see that role assignments are actually mappings of vertices and relations. That is $r \colon V \to W$ defines uniquely a mapping of relations $r_{\text{rel}} \colon \mathcal{E} \to \mathcal{F}$. Note that r_{rel} does not map edges of $\mathcal{G}$ onto edges of $\mathcal{R}$ but relations, i. e. edge sets, onto relations.

Having more then one relation, the possibilities for defining different types of role assignments explode. See [579, 471] for a large number of possibilities. We will sketch some of them.

The easiest way to translate definitions for different types of vertex partitions (see Sections 9.1, 9.2, and 9.3) to graphs with multiple relations is by the following generic definition.

Definition 9.4.3. *A role assignment* $r \colon V \to W$ *is said to be of a specific type* t *for a graph* $\mathcal{G} = (V, \mathcal{E})$ *with multiple relations, if for each* $E \in \mathcal{E}$*,* r *is of type* t *for the graph* (V, E)*.*

We illustrate this for the definition of regular equivalence relations.

Definition 9.4.4 ([579]). *Let* $\mathcal{G} = (V, \mathcal{E})$ *be a graph. A role assignment* $r \colon V \to W$ *is called* regular *for* $\mathcal{G}$ *if for each* $E \in \mathcal{E}$*,* r *is regular for graph* (V, E)*.*

Besides this natural translation of role assignments from graphs to graphs with multiple relations there is a weaker form (e.g. *weak regular network homomorphism* [579]), which makes use of the mapping of relations r_{rel}.

Theorems for certain types of vertex partitions (see Sections 9.1, 9.2, and 9.3) mostly translate to the case of multiple relations if we apply Definition 9.4.3.

Next we introduce a stronger form of compatibility with multiply relations. Regular role assignments as defined in Definition 9.4.4 make sure that equivalent vertices have, in each of the graphs relations identical ties to equivalent counterparts. Sometimes it is considered as desirable that they have the same combinations of relations to equivalent counterparts. That is, if we consider the example at the beginning of this section, it matters whether an individual gives orders to someone and is the friend of another individual or whether he gives orders to a friend.

Definition 9.4.7 formalizes this. First we need some preliminary definitions:

Definition 9.4.5 ([579]). *Given a graph $\mathcal{G} = (V, \mathcal{E})$ and $u, v \in V$, we define the* bundle (of relations) *from u to v as*

$$B_{uv} = \{E \in \mathcal{E}\,;\ (u, v) \in E\}\ .$$

These bundles define a new graph with multiple relations.

Definition 9.4.6 ([191, 579]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph and $\mathcal{B}$ be the set of all non-empty bundles. For each bundle $B \in \mathcal{B}$ defines a graph with vertex set V and edge set M_B where $(u, v) \in M_B$ if and only if $B_{uv} = B$. M_B is called a* multiplex *relation induced by the graph $\mathcal{G} = (V, \mathcal{E})$. Let $\mathcal{M} = \{M_B\}_{B\in\mathcal{B}}$, then $MPX(\mathcal{G}) := (V, \mathcal{M})$ is called the* multiplex graph *of $\mathcal{G}$.*

For each pair of vertices (u, v) there is a unique bundle associated with it. This bundle may be either empty or a member of $\mathcal{B}$ (the set of all non-empty bundles). This implies that either (u, v) is a member of no M_B or has only one such multiplex relation. Thus, the multiplex graph of a graph can be viewed as a graph with a single relation, but with edge-labels. We call such a graph a *multiplex graph* [579]. That is, a multiplex graph is a graph $\mathcal{G} = (V, \mathcal{M})$ such that for each pair of relations $M_1, M_2 \in \mathcal{M}$ either $M_1 \cap M_2 = \emptyset$ or $M_1 = M_2$ holds.

For example, the multiplex graph $MPX(\mathcal{G})$ of a graph $\mathcal{G}$, is a multiplex graph.

Now we can define the type of equivalence relation which ensures that equivalent vertices have the same bundles of relations to equivalent counterparts.

Definition 9.4.7 ([191]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph with multiple relations. A role assignment $r\colon V \to W$ that is regular for $MPX(\mathcal{G})$ is called* multiplex regular *for $\mathcal{G}$.*

As in the above definition one might define *multiplex strong structural* role assignments, but one can easily verify that a strong structural role assignment on a graph (with multiple relations) is necessarily strong structural on the corresponding multiplex graph.

Remark 9.4.8. An equivalent definition of multiplex regular role assignments is given in [83]: Let $\mathcal{G} = (V, \mathcal{E})$ be a graph, where $\mathcal{E} = \{E_1, \ldots, E_p\}$. Let

$$\mathcal{M} := \left\{ \bigcap_{i \in I} E_i \, ; \; I \subseteq \{1, \ldots, p\}, \, I \neq \emptyset \right\} \; .$$

Then the regular role assignments of $(V, \mathcal{M})$ are exactly the multiplex regular role assignments of $\mathcal{G}$.

Regular role assignments of a graph are in general not multiplex regular. Regularity however is preserved in the opposite direction.

Proposition 9.4.9 ([579]). *If $\mathcal{G} = (V, \mathcal{E})$ is a graph, $C := MPX(\mathcal{G})$, and $r \colon V \to W$ a role assignment then the following holds.*

1. *If r is regular for C then it is regular for $\mathcal{G}$.*
2. *If r is strong structural for C then it is strong structural for $\mathcal{G}$.*

Proof. For the proof of 1 and 2 let $E \in \mathcal{E}$ be a relation of $\mathcal{G}$ and let $u, v, u' \in V$ with $(u, v) \in E$ and $r(u) = r(u')$. Let B_{uv} be the bundle of relations of u and v (in particular $E \in B_{uv}$) and let $M := \{(w, w') \, ; \; B_{ww'} = B_{uv}\}$ be the corresponding multiplex relation (in particular $(u, v) \in M$).

1. If we assume that r is regular for C, there exist $v' \in V$ such that $r(v') = r(v)$ and $(u', v') \in M$, in particular it is $(u', v') \in E$ which shows the out-part of regularity for $\mathcal{G}$.
2. If we assume that r is strong structural for C, then $(u', v) \in M$, in particular it is $(u', v) \in E$ which shows the out-part of the condition for r being strong structural for $\mathcal{G}$.

The in-parts are treated analogously. □

9.5 The Semigroup of a Graph

Social relations also have an indirect influence: If A and B are friends and B and C are enemies then this (probably) has some influence on the relation between A and C.

In this section we want to formalize such higher-order relations and highlight the relationship with role assignments.

The following definitions and theorems can be found, essentially, in [579], but have been generalized here to graphs with multiple relations (see Section 9.4).

Labeled paths of relations (like EnemyOfAFriend) are formalized by composition of relations; beware of the order.

Definition 9.5.1. *If Q and R are two binary relations on V then the* (Boolean) product *of Q with R is denoted by QR and defined as*

$$QR := \{(u, v) \, ; \; \exists w \in V \text{ such that } (u, w) \in Q \text{ and } (w, v) \in R\} \; .$$

Boolean multiplication of relations corresponds to Boolean multiplication of the associated adjacency matrices, where for two $\{0,1\}$ matrices A and B the Boolean product AB is defined as

$$(AB)_{ij} = \bigvee_{k=1}^{n} A_{ik} \wedge B_{kj} \quad .$$

It is also possible to define *real* multiplication of weighted relations or multiedge sets by real matrix multiplication (this has been advocated e. g., in [89]).

Definition 9.5.2. *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph (with multiple relations). Then, the* semigroup *induced by $\mathcal{G}$ is defined to be*

$$S(\mathcal{G}) := \{E_1 \dots E_k \, ; \; k \in \mathbb{N}, \, E_1, \dots, E_k \in \mathcal{E}\} \quad .$$

We also write $S(\mathcal{E})$ for $S(\mathcal{G})$.

Note that two elements in $S(\mathcal{G})$ are equal if and only if they contain the same set of ordered pairs in $V \times V$.

Furthermore, note that $S(\mathcal{G})$ is indeed a semigroup since the multiplication of relations is associative, i. e., $(AB)C = A(BC)$ holds for all relations A, B, and C.

In general, $S(\mathcal{G})$ has no neutral element, relations have no inverse and the multiplication is not commutative.

Although the length of strings in the definition of $S(\mathcal{G})$ is unbounded, $S(\mathcal{G})$ is finite since the number of its elements is bounded by $2^{(|V|^2)}$, the number of all binary relations over V.

The interesting thing about composite relations is the identities satisfied by them. For example we could imagine that on a network of individuals with two relations Friend and Enemy, the identities FriendFriend=Friend and FriendEnemy=EnemyFriend=Enemy hold. At least the fact whether these identities hold or not gives us valuable information about the network. In all cases identities exist necessarily since $S(\mathcal{G})$ is finite but the set of all strings $\{E_1 \dots E_k \, ; \; k \in \mathbb{N}, \, E_i \in \mathcal{E}\}$ is not.

Role assignments identify individuals. Thus they introduce more identities on the semigroup of the graph. The remainder of this section investigates the relationship between role assignments and the identification of relations.

A role assignment on a graph induce a mapping on the induced semigroup.

Definition 9.5.3 ([579]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph with multiple relations and $r \colon V \to W$ a role assignment. For $Q \in S(\mathcal{G})$, $r_{\mathrm{rel}}(Q)$ (compare Section 9.4) is the relation on W defined by $r_{\mathrm{rel}}(Q) := \{(r(u), r(v)) \, ; \; (u,v) \in Q\}$ called the* relation induced *by Q and r. Thus r induces a mapping r_{rel} on the semigroup $S(\mathcal{G})$.*

Note that in general $r_{\mathrm{rel}}(S(\mathcal{G}))$ is not the semigroup of the role graph of $\mathcal{G}$ over r, however, this is true if r is regular. Role assignments do not necessarily preserve composition, i. e., r_{rel} is not a semigroup homomorphism. One of the main results (see Theorem 9.5.6) of this section is that regular role assignments have this property.

Lemma 9.5.4 ([579]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph and $r \colon V \to W$ a role assignment which is regular with respect to Q and $R \in S(\mathcal{G})$. Then, $r_{\mathrm{rel}}(QR) = r_{\mathrm{rel}}(Q)r_{\mathrm{rel}}(R)$.*

Proof. Let $w, w' \in W$ with $(w, w') \in r_{\mathrm{rel}}(QR)$. By the definition of $r_{\mathrm{rel}}(QR)$ there exist $v, v' \in V$ such that $f(v) = w$, $f(v') = w'$, and $(v, v') \in QR$. Therefore there is a vertex $u \in V$ with $(v, u) \in Q$ and $(u, v') \in R$ implying $(w, r(u)) \in r_{\mathrm{rel}}(Q)$ and $(r(c), w') \in r_{\mathrm{rel}}(R)$, whence $(w, w') \in r_{\mathrm{rel}}(Q)r_{\mathrm{rel}}(R)$. We conclude $r_{\mathrm{rel}}(QR) \subseteq r_{\mathrm{rel}}(Q)r_{\mathrm{rel}}(R)$. Note that this holds without the assumption of r being regular.

Conversely, let $w, w' \in W$ with $(w, w') \in r_{\mathrm{rel}}(Q)r_{\mathrm{rel}}(R)$. Then there is a $z \in W$ such that $(w, z) \in r_{\mathrm{rel}}(Q)$ and $(z, w') \in r_{\mathrm{rel}}(R)$. By the definition of r_{rel} there are $v, v', u_1, u_2 \in V$ with $r(v) = w$, $r(v') = w'$, $r(u_1) = r(u_2) = z$, $(v, u_1) \in Q$, and $(u_2, v') \in R$. Since r is regular and $r(u_1) = r(u_2)$ there is a vertex $v'' \in V$ with $r(v'') = f(v')$ and $(u_1, v'') \in R$. It follows that $(v, v'') \in QR$ whence $(w, w') = (r(v), r(v'')) \in r_{\mathrm{rel}}(QR)$, implying $r_{\mathrm{rel}}(Q)r_{\mathrm{rel}}(R) \subseteq r_{\mathrm{rel}}(QR)$. □

The next theorem shows that regular or strong structural on the set of generator relations $\mathcal{E}$ implies regular resp. strong structural on the semigroup $S(\mathcal{E})$. This is the second step in proving Theorem 9.5.6.

Theorem 9.5.5 ([579]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph. If $r \colon V \to W$ is regular (strong structural) with respect to $\mathcal{E}$ then r is regular (strong structural) for any relation in $S(\mathcal{G})$.*

Proof. By induction on the string length of a relation in $S(\mathcal{G})$ written as a product of generating relations (see definition 9.5.2), it suffices to show that if r is regular (strong structural) with respect to two relations $Q, R \in S(\mathcal{G})$, then it is regular (strong structural) for the product QR. So let $Q, R \in S(\mathcal{G})$ be two relations and $u, v \in V$ such that $(r(u), r(v)) \in r_{\mathrm{rel}}(QR)$. By Lemma 9.5.4, this implies $(r(u), r(v)) \in r_{\mathrm{rel}}(Q)r_{\mathrm{rel}}(R)$, whence there is a $w \in W$ such that $(r(u), w) \in r_{\mathrm{rel}}(Q)$ and $(w, r(v)) \in r_{\mathrm{rel}}(R)$. Since r is surjective, there exists $u_0 \in V$ with $r(u_0) = w$, and it is $(r(u), r(u_0)) \in r_{\mathrm{rel}}(Q)$ and $(r(u_0), r(v)) \in r_{\mathrm{rel}}(R)$.

Now, suppose that r is *regular* with respect to Q and R. We have to show the existence of $c, d \in V$ such that $(c, v) \in QR$, $(u, d) \in QR$, $r(c) = r(u)$ and $r(d) = r(v)$. Since r is regular with respect to Q and $(r(u), r(u_0)) \in r_{\mathrm{rel}}(Q)$ there exists $u_1 \in V$ such that $r(u_1) = r(u_0)$ and $(u, u_1) \in Q$. Similarly, since r is regular with respect to R and $(r(u_0), r(v)) \in r_{\mathrm{rel}}(R)$, there exists $d \in V$ such that $r(d) = r(v)$, and $(u_1, d) \in R$. Since $(u, u_1) \in Q$ and $(u_1, d) \in R$ it follows $(u, d) \in QR$, which is the first half of what we have to show. The proof of the second half can be done along the same lines.

Now, suppose that f is *strong structural* with respect to Q and R. Then $(r(u), r(u_0)) \in r_{\text{rel}}(Q)$ and $(r(u_0), r(v)) \in r_{\text{rel}}(R)$ immediately implies $(u, u_0) \in Q$ and $(u_0, v) \in R$, whence $(u, v) \in QR$. □

The next theorem might be seen as the main result of this section. It states that regular role assignments induce homomorphisms on the induced semigroups.

Theorem 9.5.6 ([579]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph with multiple relations. If $r \colon V \to W$ is a regular role assignment with role graph $\mathcal{R}$, then $r_{\text{rel}} \colon S(\mathcal{G}) \to S(\mathcal{R})$ is a surjective semigroup homomorphism.*

Proof. We know from Lemma 9.5.4 that the identity $r_{\text{rel}}(QR) = r_{\text{rel}}(Q) r_{\text{rel}}(R)$ holds whenever r is regular with respect to Q and R. Theorem 9.5.5 states that r is regular with respect to all relations in $S(\mathcal{G})$. Thus the image of $S(\mathcal{G})$ under r_{rel} is equal to $S(\mathcal{R})$ (the images of the generator relations $\mathcal{E}$ are the generator relations of the semigroup of the role graph $S(\mathcal{R})$) and r_{rel} is a semigroup homomorphism. □

The condition that r be regular, is not necessary for r_{rel} being a semigroup homomorphism. Kim and Roush [355] gave a more general sufficient condition. Also compare [471].

The next theorem shows that the role graph of a strong structural role assignment has the same semigroup as the original graph.

Theorem 9.5.7 ([579]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph with multiple relations. If $r \colon V \to W$ is a strong structural role assignment with role graph $\mathcal{R}$, then $r_{\text{rel}} \colon S(\mathcal{G}) \to S(\mathcal{R})$ is a semigroup isomorphism.*

Proof. By Theorem 9.5.6 r_{rel} is a surjective semigroup homomorphism. It remains to show that r_{rel} is injective. So let $Q, R \in S(\mathcal{G})$ with $r_{\text{rel}}(Q) = r_{\text{rel}}(R)$. Then, for all $u, v \in V$ if holds $(u, v) \in Q$ iff $(r(u), r(v)) \in r_{\text{rel}}(Q)$ (since r is strong) iff $(r(u), r(v)) \in r_{\text{rel}}(R)$ iff $(u, v) \in R$ (since r is strong). □

Do Semigroup-Homomorphisms Reduce Networks? The above theorems give the idea to an alternative approach to find role assignments: In Theorem 9.5.6 it has been shown that role assignments introduce new identities on the semigroup of (generator and compound) relations of a network. Conversely, one could impose identities on relations that are almost satisfied, or that are considered to be reasonable. Now the interesting question is: *Does identification of relations imply identification of vertices of the graph which generated the semigroup?* (See [73].)

That is, given a graph $\mathcal{G}$ with semigroup $S(\mathcal{G})$ and a surjective semigroup homomorphism $S(\mathcal{G}) \to S'$ onto some semigroup S', *is there a graph $\mathcal{G}'$ and a graph homomorphism $\mathcal{G} \to \mathcal{G}'$ such that S' is the semigroup generated by $\mathcal{G}'$?*

This would be the counterpart of Theorem 9.5.6, which states that role assignments on graphs induce, under the condition of regularity, reductions of the induced semigroups, (i. e., surjective semigroup homomorphisms).

The answer is in general *no*, simply for the reason that not every semigroup is a semigroup of relations. But *under what conditions on S' and on the semigroup* *homomorphism would we get a meaningful role graph and a meaningful role assignment?*

Although the question is open for the general case some examples can be found in [89] and [471].

9.5.1 Winship-Pattison Role Equivalence

The condition for regular equivalent vertices is: *equivalent vertices have the same ties to equivalent counterparts.* In this section the phrase *to equivalent counterparts* is replaced by the weaker requirement *to some vertices.* As mentioned in Remark 9.5.9 the four equivalences defined in this section, are special cases of relative regular equivalence (see Section 9.3.4).

Definition 9.5.8. *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph and $\sim$ an equivalence on V. Then $\sim$ is said to be a* weak role equivalence *for $\mathcal{G}$ if for all $u, v, w \in V$ and $E \in \mathcal{E}$, $u \sim v$ implies both*

– *uRw implies there exists x such that vRx,*
– *wRu implies there exists x such that xRv.*

Note that in contrast to the definition of regular equivalence one does not consider the role of x. So weak role-equivalent vertices don't share the same relations *to equivalent counterparts*, but they only share the same relations. If the graph has one single relation, the maximal weak role equivalence is simply the partition into isolates, sinks, sources, and vertices with positive in- and out-degree.

The indifference in regard to the role of adjacent vertices makes weak role equivalence a much weaker requirement than e. g., regular or strong structural equivalences.

Weak role equivalence could have been defined using relative regular equivalence (see Section 9.3.4).

Remark 9.5.9. Weak role equivalences are exactly the equivalences which are regular relative to the complete partition. This remark immediately generalizes to the next three definitions.

Weak role equivalence can be tightened in two directions: to include multiplexity, which leads to Definition 9.5.11, or to include composition of relations, which leads to Definition 9.5.10.

Definition 9.5.10. *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph, $S := S(\mathcal{G})$ its semigroup, and $\sim$ an equivalence on V. Then $\sim$ is called a* compositional equivalence *of $\mathcal{G}$ if it is a weak role equivalence of (V, S) (see Definition 9.5.8).*

Note that in contrast to regular equivalences, where an equivalence is regular with respect to $\mathcal{E}$ if and only if it is regular with respect to $S(\mathcal{E})$, it makes a difference whether we require $\sim$ to be a weak role equivalence of $\mathcal{G}$ or of (V, S). Compositional equivalences are weak role equivalences.

Definition 9.5.11 ([579]). *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph, $C = (V, \mathcal{M}) := MPX(\mathcal{G})$ its multiplex graph (see Definition 9.4.6) and $\sim$ an equivalence on V. Then, $\sim$ is called a* bundle equivalence *of $\mathcal{G}$ if it is a weak role equivalence (see Definition 9.5.8) of C.*

Bundle equivalences are weak role equivalences.

Winship-Pattison role equivalence is most often defined in terms of the *role-set* of an actor (see [471, p. 79ff]): Two actors are equivalent if they have the same role-sets (also compare [82, p. 81]). We restate the definitions given there in our terminology.

Definition 9.5.12. *Let $\mathcal{G} = (V, \mathcal{E})$ be a graph. An equivalence relation $\sim$ on V is called a* local role equivalence *or* Winship-Pattison role equivalence *if $\sim$ is a bundle equivalence (see Definition 9.5.11) of the graph $(V, S(\mathcal{G}))$.*

Local role equivalences are both bundle and compositional equivalences. Local role equivalences are, in general, not regular, which immediately implies the same for the three other (weaker) equivalences defined in this section: Let vertices u and v be connected by a bidirected edge and v have an out-going edge to a third vertex w. Then u and v are locally role equivalent but not regularly equivalent.

Conclusion. The semigroup of a graph is a possibility to describe the interaction of multiple and compound relations. An idea to use identification of relations in order to get role assignments has been sketched. This approach seems to be rather hard, both theoretically and computationally.

9.6 Chapter Notes

Vertex partitions that yield role assignments have first been introduced by Lorrain and White [394], who defined structural equivalence.

Sailer [501] pointed out that structural equivalence is to restrictive to meet the intuitive notion of social role. He proposed that actors play the same role if they are connected to *role-equivalent* actors (in contrast to *identical* actors, as structural equivalence demands). His idea of structural relatedness has been formalized as regular equivalence by White and Reitz in the seminal paper [579]. In this work, they gave a unified treatment of structural, regular, and other equivalences for graphs with single or multiple relations. Furthermore, they developed conditions for graph homomorphisms to induce (structural or regular) vertex partitions and to be compatible with the composition of relations.

Borgatti and Everett [82, 83, 190, 191] established many properties of the set of regular equivalences, including lattice structure, and developed the algorithm CATREGE to compute the maximal regular equivalence of a graph. Furthermore they introduced other types of vertex partitions to define roles in graphs. Boyd and Everett [90] further clarified the lattice structure and defined relative regular equivalence.

Marx and Masuch [408] commented that regular equivalence is already known, under the name of bisimulation in computer science. Their report has been the reason that we found the algorithm of Paige and Tarjan [459], which can compute the maximal regular equivalence and is much faster than CATREGE.

Roberts and Sheng [493] first showed that there are $\mathcal{NP}$-complete problems stemming from regular role assignments. A more complete treatment is from Fiala and Paulusma [209].

Role assignments for graphs with multiple and composite relations are already treated in [394, 579]. The possibilities to define role assignments in graphs with multiple relations are abundant. We could sketch only few of them in this chapter. Additional reading is, e. g., Kim and Roush [355] and Pattison [471] who found many conditions for vertex partitions to be compatible with the composition of relations. In the latter book, the algebraic structure of semigroups of relations is presented in detail. Boyd [89] advocated the use of real matrix multiplication to define semigroups stemming from graphs. These semigroups often admit sophisticated decompositions, which in turn, induce decompositions or reductions of the graphs that generated these semigroups.

In order to be able to deal with the irregularities of empirical networks, a formalization of role assignment must – in addition to choosing the right compatibility criterion – provide some kind of relaxation. (See Wasserman and Faust [569] for a more detailed explanation.) Relaxation has not been treated in this chapter, which has been focused on the 'ideal' case of vertex partitions that satisfy strictly the different compatibility constraints. Possibilities to relax structural equivalence, optimizational approaches for regular equivalence, and stochastic methods for role assignments are presented in Chapter 10 about blockmodels. Brandes and Lerner [97] introduced a relaxation of equitable partitions to provide a framework for role assignments that are tolerant towards irregularities.