1 Idea

A set of vertices $D_\alpha \subseteq V$ is a super dominion for player α in VPG G iff D_α is a dominion for player α in $G_{|c}$ for any $c \in \mathfrak{C}$.

Define $\alpha{-}Gen : VPG \to PG$ such that $\alpha{-}Gen(V, V_0, V_1, E, \Omega, \mathfrak{C}, \theta) = (V, V_0, V_1, E', \Omega)$ with $E' = \{(v, w) \in E \mid v \in V_{\overline{\alpha}} \vee \theta(v, w) = \mathfrak{C}\}$. Note that the resulting PG might contain deadlocks (ie. finite plays are possible).

Theorem 1.1. *W_α in $\alpha{-}Gen(G)$ is a super dominion in G for player α.*

2 Alg

Define $\cap : VPG \to \mathcal{P}(\mathfrak{C}) \to VPG$
$(V, V_0, V_1, E, \Omega, \mathfrak{C}, \theta) \cap \mathfrak{C}' = (V, V_0, V_1, E', \Omega, \mathfrak{C}', \theta')$ such that:
$\mathfrak{C}' = \mathfrak{C} \cap \mathfrak{C}'$
$\theta'(e) = \theta(e) \cap \mathfrak{C}'$ for every $e \in E$
$E' = \{e \in E \mid \theta'(e) \neq \emptyset\}$

Let DASolve be a dominion aware parity game solve algorithm, such an algorithm uses the dominions that are already known. Given DASolve we can solve a VPG using the following algorithm:

Algorithm 1 SuperDominionMBR(G, D_0, D_1)

```
if |𝕮| = 1 then
    (W_0, W_1) ← DASolve(G, D_0, D_1)
    return (𝕮 × W_0, 𝕮 × W_1)
end if
(D'_0, −) ← DASolve(0−Gen(G), D_0, D_1)
(−, D'_1) ← DASolve(1−Gen(G), D_0, D_1)
if D'_0 ∪ D'_1 = V then
    return (𝕮 × D'_0, 𝕮 × D'_1)
end if
(𝕮^a, 𝕮^b) ← partition 𝕮 in non-empty parts
(W_0^a, W_1^a) ← SuperDominionMBR(G ∩ 𝕮^a, D'_0, D'_1)
(W_0^b, W_1^b) ← SuperDominionMBR(G ∩ 𝕮^b, D'_0, D'_1)
W_0 ← W_0^a ∪ W_0^b
W_1 ← W_1^a ∪ W_1^b
return (W_0, W_1)
```

Any parity game algorithm Solve can be transformed in a dominion aware algorithm by using the following transformation:

Algorithm 2 DASolveGeneric(G, D_0, D_1)

```
for α ∈ {0, 1} do
    Replace D_α in G with a single vertex d_α with priority α and a selfloop. All edges going from D_α are
    removed. Edges going from v ∈ V_α to D_α are replaced with (v, d_α). Edges going from v ∈ V_ᾱ to D_α are
    removed.
end for
(W_0, W_1) ← Solve(G)
return (W_0\{d_0} ∪ D_0, W_1\{d_1} ∪ D_1)
```

3 Fixpoint iteration

For some algorithms, specifically iteration algorithms, we can use the internal structure instead of using super dominions. For example for the fixpoint iterator we can use the values of the variables.

Algorithm 3 FIXPOINTITERMBR$(G, (Z_{n-1}, \dots, Z_0))$

if $|\mathfrak{C}| = 1$ **then**
  $(Z'_{n-1}, \dots, Z'_0) \leftarrow$ fixpoint iterate over game G starting at $Z_{n-1}, \dots, Z_0$
  return $(\mathfrak{C} \times Z'_{n-1}, \mathfrak{C} \times (V \backslash Z'_{n-1}))$
end if
$(Z^0_{n-1}, \dots, Z^0_0) \leftarrow$ fixpoint iterate over game $0{-}Gen(G)$ starting at $Z_{n-1}, \dots, Z_0$
$(Z^1_{n-1}, \dots, Z^1_0) \leftarrow$ fixpoint iterate over game $1{-}Gen(G)$ starting at $Z_{n-1}, \dots, Z_0$
$Z'_i \leftarrow Z_i^{i-1 \mod 2}$ for every $i \in [n-1]$
$(\mathfrak{C}^a, \mathfrak{C}^b) \leftarrow$ partition $\mathfrak{C}$ in non-empty parts
$(W^a_0, W^a_1) \leftarrow$ FIXPOINTITERMBR$(G \cap \mathfrak{C}^a, (Z'_{n-1}, \dots, Z'_0))$
$(W^b_0, W^b_1) \leftarrow$ FIXPOINTITERMBR$(G \cap \mathfrak{C}^b, (Z'_{n-1}, \dots, Z'_0))$
$W_0 \leftarrow W^a_0 \cup W^b_0$
$W_1 \leftarrow W^a_1 \cup W^b_1$
return (W_0, W_1)

Fixpoint equation:

$$\begin{aligned} F_0(Z_{n-1}, \dots, Z_1) = \{x \in V \mid & \\ & (x \in V_0 \implies \exists_y : ((x,y) \in E \land y \in Z_{\Omega(y)})) \\ & \land \\ & (x \in V_1 \implies \forall_y : ((x,y) \in E \implies y \in Z_{\Omega(y)})) \\ & \} \end{aligned}$$

$$W_0 = \sigma Z_{n-1} \dots \mu Z_1 . \nu Z_0 . F_0(Z_{n-1}, \dots, Z_1)$$

Every Z_i has fixpoint operator ν if i is even and μ if i is odd.
Claim: For game $0{-}Gen(G \cap \mathfrak{C}')$ we have that F_0 increases when $\mathfrak{C}'$ decreases.
Claim: For game $1{-}Gen(G \cap \mathfrak{C}')$ we have that F_0 decreases when $\mathfrak{C}'$ decreases.
Note that fixpoint iteration works for games with deadlocks.

4 Small progress measure

For game $0{-}Gen(G \cup \mathfrak{C}')$ it holds that the progress measure is larger or smaller when $\mathfrak{C}'$ decreases. I think.