BUGGER

FW = Folge Wand
W = Warte
FaW = Finde andere Wand

zb_u_ns = Zielfeld belegt u. nicht Start?
zf = Zielfeld frei?
zb = Zielfeld belegt?
zf_u_ns = Zielfeld frei u. nicht Start?
zs = Zielfeld Start?
zw = Zielfeld Wand?
znw[*1)] = Zielfeld nicht Wand?

	zb_u_ns	**zf**	**zb**	**zf_u_ns**	**zs**	**zw**	**znw**
➔ **FW**[*]	W	–	–	FW	FaW	–	–
W	–	FW	W	–	–	–	–
FaW[*)]	–	–	–	–	–	FW	FaW

[*)] Bei Zustand nochmal Event prüfen um richtiges Zielfeld gespeichert zu haben

[*1)] Da die Graphen in der Aufgabenstellung so schlecht durchdacht sind, findet der Check, ob das Zielfeld frei ist in der state_func zu FaW statt

PUSHER[1]

HB = Horizontale Bewegung
VB = Vertikale Bewegung
Z = Zerstört

zf = Zielfeld frei (Oder Roboter, aber verschiebbar)?
zw_o_pbb = Zielfeld Wand (o. block. Rob.) o. Patchbot-Breite erreicht?
zw_o_pbh = Zielfeld Wand (o. block. Rob.) o. Patchbot-Höhe erreicht?
zg = Zielfeld gefährlich?

	zf	**zw_o_pbb**	**zw_o_pbh**	**zg**
➔ **HB**	HB	VB	–	Z
VB	VB	–	HB	Z
Z	–	–	–	–

DIGGER[1]

HB = Horizontale Bewegung
VB = Vertikale Bewegung
Z = Zerstört

zf = Zielfeld frei (Oder Roboter, aber verschiebbar)?
zw_o_pbb = Zielfeld Wand (o. block. Rob.) o. Patchbot-Breite erreicht?
zw_o_pbh = Zielfeld Wand (o. block. Rob.) o. Patchbot-Höhe erreicht?
zg = Zielfeld gefährlich?

	zf	**zw_o_pbb**	**zw_o_pbh**	**zg**

[1] Hier gibt es weitere Events, die auf den gleichen Zustand zeigen, z.B. Bewegung oder Verschieben (lt. Aufgabenstellung). Es werden hier zur Vereinfachung und zur Kompakthaltung der States und Events etwaige Nebeneffekte (z.B. Verschieben oder Graben) mit der Bewegung an sich zusammengefasst. Das v.a., da die Aufgabenstellung meiner Meinung nach sehr inkonsistent formuliert ist.

➔ **HB**	HB	VB	–	Z
VB	VB	–	HB	Z
Z	–	–	–	–

SWIMMER[2]

HB = Horizontale Bewegung
VB = Vertikale Bewegung
Z = Zerstört

zf_o_w = Zielfeld frei (Oder Roboter, aber verschiebbar) o. Wasser?
zw_o_pbb = Zielfeld Wand (o. block. Rob.) o. Patchbot-Breite erreicht?
zw_o_pbh = Zielfeld Wand (o. block. Rob.) o. Patchbot-Höhe erreicht?
zg = Zielfeld gefährlich?

	zf_o_w	**zw_o_pbb**	**zw_o_pbh**	**zg**
➔ **HB**	HB	VB	–	Z
VB	VB	–	HB	Z
Z	–	–	–	–

PUSHING ROBOT[2]

(Pushing Robot sind alle „pushing robots" zusammengefasst. So wurden sie auch implementiert. Roboter-spezifische Aktionen (z.B. schwimmen und graben, wurden dort konditional mit eingebaut.)

HB = Horizontale Bewegung
VB = Vertikale Bewegung
Z = Zerstört

zf = Zielfeld frei (Oder Roboter, aber verschiebbar) o. Wasser?
zw_o_pbb = Zielfeld Wand (o. block. Rob.) o. Patchbot-Breite erreicht?
zw_o_pbh = Zielfeld Wand (o. block. Rob.) o. Patchbot-Höhe erreicht?
zg = Zielfeld gefährlich?

	zf_o_w	**zw_o_pbb**	**zw_o_pbh**	**zg**
➔ **HB**	HB	VB	–	Z
VB	VB	–	HB	Z
Z	–	–	–	–

[2] s. Fußnote 1

FOLLOWER[3]

V = Verfolge
W = Warte

pb_is = Patchbot in Sicht und erreichbar?
pb_as = Patchbot außer Sicht oder nicht erreichbar?

	pb_is	**pb_as**
➔ **W**	V	W
V	V	W

HUNTER[3]

V = Verfolge
W = Warte
J = Jage

pb_is = Patchbot in Sicht und erreichbar?
pb_as = Patchbot außer Sicht oder nicht erreichbar?

	pb_is	**pb_as**
➔ **W**	V	-
V	V	J
J	V	J

SNIFFER

V = Verfolge
W = Warte

pe = Patchbot erreichbar?
pne = Patchbot nicht erreichbar?

	pe	**pne**
➔ **W**	V	W
V	V	W

Follower, Hunter und Sniffer zusammenzufassen gestaltet sich etwas schwieriger, zumindest als Tabelle hier. Bei der Implementierung bietet es sich an, um trotzdem alle drei FSMs in einer Klasse umzusetzen, eine map<robot_type, fsm-table> zu verwalten.

[3] Dass sich Patchbot „in Sicht“ befindet, heißt hier, dass jedes Tile t mit x und y zwischen Patchbot pb mit x und y und Gegener g mit x und y: $x_t : x_{pb} \leq x_t \leq x_g \ \wedge \ y_t : y_t = \frac{y_g - y_{pb}}{x_g - x_{pb}} x_t + y_g - \frac{y_g - y_{pb}}{x_g - x_{pb}} x_g \vee x_t : x_t = \frac{\left(y_t - y_g - \frac{y_g - y_{pb}}{x_g - x_{pb}} x_g\right)}{\frac{y_g - y_{pb}}{x_g - x_{pb}}} \wedge y_t : y_{pb} \leq y_t \leq y_g$ nicht vom Typ Wand ist. Typ Wand sei hier terrain::CONCRETE_WALL, terrain::ROCK_WALL, terrain::AUTOMATIC_DOOR, terrain::MANUAL_DOOR oder terrain::SERET_PASSAGE. Wohl gemerkt sind die Türen geschlossen. Wenn das Ergebnis der o.g. Funktionen direkt auf ein Tile fällt, d.h. $y_t = \lfloor y\_t \rfloor \vee x_t = \lfloor x_t \rfloor$, je nachdem ob $\Delta x >$ Δy, werden auch die diagonale nächsten Tiles mit geprüft (also x-1, y und x, y-1).